intel

Embedded Controller Handbook

Volume I 8-Bit



Intel the Microcomputer Compa.

When Intel invented the microprocessor in 1971, it created the era o. microcomputers. Whether used as microcontrollers in automobiles or microwave ovens, or as personal computers or supercomputers, Intel's microcomputers have always offered leading-edge technology. In the second half of the 1980s, Intel architectures have held at least a 75% market share of microprocessors at 16 bits and above. Intel continues to strive for the highest standards in memory, microcomputer components, modules, and systems to give its customers the best possible competitive advantages.

EMBEDDED CONTROLLER HANDBOOK

1988

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and may only be used to identify Intel Products:

Above, BITBUS, COMMputer, CREDIT, Data Pipeline, FASTPATH, GENIUS, i, $\hat{i}$, ICE, iCEL, iCS, iDBP, iDIS, I^2ICE, iLBX, i_m, iMDDX, iMMX, Inboard, Insite, Intel, intel, intelBOS, Intel Certified, Intelevision, intelligent Identifier, intelligent Programming, Intellec, Intellink, iOSP, iPDS, iPSC, iRMK, iRMX, iSBC, iSBX, iSDM, iSXM, KEPROM, Library Manager, MAP-NET, MCS, Megachassis, MICROMAINFRAME, MULTIBUS, MULTICHANNEL, MULTIMODULE, MultiSERVER, ONCE, OpenNET, OTP, PC-BUBBLE, Plug-A-Bubble, PROMPT, Promware, QUEST, QueX, Quick-Pulse Programming, Ripplemode, RMX/80, RUPI, Seamless, SLD, SugarCube, SupportNET, UPI, and VLSiCEL, and the combination of ICE, iCS, iRMX, iSBC, iSBX, iSXM, MCS, or UPI and a numerical suffix, 4-SITE.

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

*MULTIBUS is a patented Intel bus.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Distribution
Mail Stop SC6-59
3065 Bowers Avenue
Santa Clara, CA 95051

© INTEL CORPORATION 1987

Table of Contents

Alphanumeric Index viii
Preface xiii

8-BIT PRODUCTS

MCS®-48 FAMILY

Chapter 1
MCS®-48 Single Component System 1-1

Chapter 2
MCS®-48 Expanded System 2-1

Chapter 3
MCS®-48 Instruction Set 3-1

Chapter 4
DATA SHEETS
8243 MCS®-48 Input/Output Expander 4-1
P8748H/P8749H/8048AH/8035AHL/8049AH/8039AHL/8050AH/8040AHL HMOS Single-Component 8-Bit Production Microcontroller 4-8
D8748H/8749H HMOS-E Single-Component 8-Bit Microcomputer 4-21
MCS®-48 Express 4-33
MCS®-48 INDEX 4-36

MCS®-51 FAMILY

Chapter 5
MCS®-51 Architectural Overview 5-1

Chapter 6
Hardware Description of the 8051, 8052 and 80C51 6-1

Chapter 7
Hardware Description of the 83C51FA (83C252) 7-1

Chapter 8
Hardware Description of the 83C152 8-1

Chapter 9
MCS®-51 Programmer's Guide and Instruction Set 9-1

Chapter 10
DATA SHEETS
8031/8051/8031AH/8051AH/8032AH/8052AH/8751H/8751H-8 8-Bit HMOS and HMOS EPROM Microcontrollers 10-1
8051AHP 8-Bit Control-Oriented Microcontroller with Protected ROM 10-15
8031AH/8051AH/8032AH/8052AH/8751H/8751H-8 Express 10-25
8751BH 8-Bit HMOS EPROM Microcontroller 10-27
8032BH/8052BH 8-Bit HMOS Microcontrollers 10-38
8752BH 8-Bit HMOS EPROM Microcontroller 10-46
80C31BH/80C51BH 8-Bit CHMOS Microcontrollers 10-57
80C31BH/80C51BH Express 10-69
87C51 8-Bit CHMOS EPROM Microcontroller 10-71
87C51 Express 10-84
87C51FA (87C252) CHMOS Single-Chip 8-Bit Microcontroller 10-87
83C152A/80C152A Universal Communications Controller 10-102
80C152JA/83C152JA/80C152JB Universal Communications Controller 10-117
27C64/87C64 64K (8K x 8) CHMOS Production and UV Erasable PROMs 10-133
87C257 256K (32K x 8) CHMOS UV Erasable PROM 10-146
UPI™-452 CHMOS Programmable I/O Processor 10-157
APPLICATION NOTES
AP-70 Using the Intel MCS®-51 Boolean Processing Capabilities 10-211
AP-125 Designing Microcontroller Systems for Electrically Noisy Environments 10-256
AP-155 Oscillators for Microcontrollers 10-278

Table of Contents (Continued)

AP-252 Designing with the 80C51BH 10-310
AP-281 UPI™-452 Accelerates iAPX 286 Bus Performance 10-334
ARTICLE REPRINTS
AR-409 Increased Functions in Chip Result in Lighter, Less Costly Portable Computer 10-354
AR-517 Using the 8051 Microcontroller with Resonant Transducers 10-359
DEVELOPMENT SUPPORT TOOLS
8051 Software Packages 10-364
iDCX 51 Distributed Control Executive 10-372
ICE™ 5100/252 In-Circuit Emulator for the MCS®-51 Family 10-380
MCS®-51 INDEX 10-390
80C152 INDEX 10-392

THE RUPI™ FAMILY

Chapter 11
The RUPI™-44 Family 11-1

Chapter 12
8044 Architecture 12-1

Chapter 13
8044 Serial Interface 13-1

Chapter 14
8044 Application Examples 14-1

Chapter 15
DATA SHEET
8044AH/8344AH/8744H High Performance 8-Bit Microcontroller with On-Chip Serial Communication Controller 15-1
APPLICATION NOTE
AP-283 Flexibility in Frame Size with the 8044 15-27
ARTICLE REPRINT
AR-307 Microcontroller with Integrated High Performance Communications Interface 15-57
DEVELOPMENT SUPPORT TOOLS
ICE™5100/044 In-Circuit Emulator for the RUPI™-44 Family 15-66

MCS®-80/85 FAMILY

Chapter 16
DATA SHEETS
8080A/8080A-1/8080A-2 8-Bit N-Channel Microprocessor 16-1
8085AH/8085AH-2/8085AH-1 8-Bit HMOS Microprocessors 16-11
8155H/8156H/8155H-2/8156H-2 2048-Bit Static HMOS RAM with I/O Ports and Timer 16-31
8185/8185-2 1024 x 8-Bit Static RAM for MCS®-85 16-45
8224 Clock Generator and Driver for 8080A CPU 16-50
8228 System Controller and Bus Driver for 8080A CPU 16-55
8755A 16,384-Bit EPROM with I/O 16-59

16-BIT PRODUCTS

MCS®-96 FAMILY

Chapter 17
MCS®-96 Architectural Overview 17-1

Chapter 18
MCS®-96 Instruction Set 18-1

Chapter 19
MCS®-96 Hardware Design Information 19-1

Table of Contents (Continued)

Chapter 20

80C196KA Architectural Overview 20-1

Chapter 21

DATA SHEETS

809XBH/839XBH/879XBH with 8 or 16-Bit External Bus 21-1

809XBH-10 Advanced 16-Bit Microcontroller with 8 or 16-Bit External Bus 21-44

809X-90, 839X-90 21-59

809XBH/839XBH/879XBH Express 21-78

809X-90, 839X-90 Express 21-87

80C196KA 16-Bit High Performance CHMOS Microcontroller 21-92

APPLICATION NOTES

AP-248 Using the 8096 21-119

AP-275 An FFT Algorithm for MCS®-96 Products Including Supporting Routines and Examples 21-222

DEVELOPMENT SUPPORT TOOLS

MCS®-96 Software Development Packages 21-297

iDCX 96 Distributed Control Executive 21-307

iSBE-96 Development Kit Single Board Emulator and Assembler for MCS®-96 21-315

VLSICE™-96 In-Circuit Emulator for the MCS®-96 21-323

ICE™-196 Real-Time Transparent 80C196 In-Circuit Emulator 21-333

MCS®-96 INDEX 21-335

80C196 INDEX 21-341

80186 FAMILY

Chapter 22

DATA SHEETS

80186 High Integration 16-Bit Microprocessor 22-1

80C186 High Integration 16-Bit Microprocessor 22-53

80188 High Integration 16-Bit Microprocessor 22-111

80C188 High Integration 16-Bit Microprocessor 22-165

82188 Integrated Bus Controller for 8086, 8088, 80186, 80188 Processors 22-225

APPLICATION NOTES

AP-186 Introduction to the 80186 Microprocessor 22-241

AP-258 High Speed Numerics with the 80186, 80188 and 8087 22-316

AP-286 80186/188 Interface to Intel Microcontrollers 22-332

DEVELOPMENT SUPPORT TOOLS

8086/80186 Software Packages 22-362

VAX/VMS Resident 8086/8088/80186 Software Development Packages 22-383

8087 Support Library 22-391

80287 Support Library 22-395

iPAT Performance Analysis Tool 22-399

I²ICE™ Integrated Instrumentation and In-Circuit Emulation System 22-412

ICE™-186 In-Circuit Emulator 22-424

Alphanumeric Index

27C64/87C64 64K (8K x 8) CHMOS Production and UV Erasable PROMs 10-133
80C152JA/83C152JA/80C152JB Universal Communications Controller 10-117
80C186 High Integration 16-Bit Microprocessor 22-53
80C188 High Integration 16-Bit Microprocessor 22-165
80C196KA Architectural Overview 20-1
80C196KA 16-Bit High Performance CHMOS Microcontroller 21-92
80C31BH/80C51BH Express 10-69
80C31BH/80C51BH 8-Bit CHMOS Microcontrollers 10-57
80186 High Integration 16-Bit Microprocessor 22-1
80188 High Integration 16-Bit Microprocessor 22-111
80287 Support Library 22-395
8031/8051/8031AH/8051AH/8032AH/8052AH/8751H/8751H-8 8-Bit HMOS and HMOS EPROM Microcontrollers 10-1
8031AH/8051AH/8032AH/8052AH/8751H/8751H-8 Express 10-25
8032BH/8052BH 8-Bit HMOS Microcontrollers 10-38
8044 Application Examples 14-1
8044 Architecture 12-1
8044 Serial Interface 13-1
8044AH/8344AH/8744H High Performance 8-Bit Microcontroller with On-Chip Serial Communication Controller 15-1
8051 Software Packages 10-364
8051AHP 8-Bit Control-Oriented Microcontroller with Protected ROM 10-15
8080A/8080A-1/8080A-2 8-Bit N-Channel Microprocessor 16-1
8085AH/8085AH-2/8085AH-1 8-Bit HMOS Microprocessors 16-11
8086/80186 Software Packages 22-362
8087 Support Library 22-391
809X-90, 839X-90 21-59
809X-90, 839X-90 Express 21-87
809XBH-10 Advanced 16-Bit Microcontroller with 8 or 16-Bit External Bus 21-44
809XBH/839XBH/879XBH with 8 or 16-Bit External Bus 21-1
809XBH/839XBH/879XBH Express 21-78
8155H/8156H/8155H-2/8156H-2 2048-Bit Static HMOS RAM with I/O Ports and Timer 16-31
8185/8185-2 1024 x 8-Bit Static RAM for MCS®-85 16-45
82188 Integrated Bus Controller for 8086, 8088, 80186, 80188 Processors 22-225
8224 Clock Generator and Driver for 8080A CPU 16-50
8228 System Controller and Bus Driver for 8080A CPU 16-55
8243 MCS®-48 Input/Output Expander 4-1
83C152A/80C152A Universal Communications Controller 10-102
87C257 256K (32K x 8) CHMOS UV Erasable PROM 10-146
87C51 Express 10-84
87C51 8-Bit CHMOS EPROM Microcontroller 10-71
87C51FA (87C252) CHMOS Single-Chip 8-Bit Microcontroller 10-87
8751BH 8-Bit HMOS EPROM Microcontroller 10-27
8752BH 8-Bit HMOS EPROM Microcontroller 10-46
8755A 16,384-Bit EPROM with I/O 16-59
AP-275 An FFT Algorithm for MCS®-96 Products Including Supporting Routines and Examples 21-222
AP-125 Designing Microcontroller Systems for Electrically Noisy Environments 10-256
AP-155 Oscillators for Microcontrollers 10-278
AP-186 Introduction to the 80186 Microprocessor 22-241
AP-248 Using the 8096 21-119
AP-252 Designing with the 80C51BH 10-310
AP-258 High Speed Numerics with the 80186, 80188 and 8087 22-316
AP-281 UPI™-452 Accelerates iAPX 286 Bus Performance 10-334

Alphanumeric Index (Continued)

AP-283 Flexibility in Frame Size with the 8044 15-27
AP-286 80186/188 Interface to Intel Microcontrollers 22-332
AP-70 Using the Intel MCS®-51 Boolean Processing Capabilities 10-211
AR-409 Increased Functions in Chip Result in Lighter, Less Costly Portable Computer 10-354
AR-307 Microcontroller with Integrated High Performance Communications Interface 15-57
AR-517 Using the 8051 Microcontroller with Resonant Transducers 10-359
D8748H/8749H HMOS-E Single-Component 8-Bit Microcomputer 4-21
Hardware Description of the 8051, 8052 and 80C51 6-1
Hardware Description of the 83C152 8-1
Hardware Description of the 83C51FA (83C252) 7-1
I²ICE™ Integrated Instrumentation and In-Circuit Emulation System 22-412
iDCX 51 Distributed Control Executive 10-372
iDCX 96 Distributed Control Executive 21-307
iPAT Performance Analysis Tool 22-399
iSBE-96 Development Kit Single Board Emulator and Assembler for MCS®-96 21-315
ICE™ 5100/252 In-Circuit Emulator for the MCS®-51 Family 10-380
ICE™-186 In-Circuit Emulator 22-424
ICE™-196 Real-Time Transparent 80C196 In-Circuit Emulator 21-333
ICE™5100/044 In-Circuit Emulator for the RUPI™-44 Family 15-66
MCS®-48 Expanded System 2-1
MCS®-48 Express 4-33
MCS®-48 Instruction Set 3-1
MCS®-48 Single Component System 1-1
MCS®-51 Architectural Overview 5-1
MCS®-96 Architectural Overview 17-1
MCS®-96 Hardware Design Information 19-1
MCS®-96 Instruction Set 18-1
MCS®-96 Software Development Packages 21-297
P8748H/P8749H/8048AH/8035AHL/8049AH/8039AHL/8050AH/8040AHL HMOS Single-Component 8-Bit Production Microcontroller 4-8
The RUPI™-44 Family 11-1
UPI™-452 CHMOS Programmable I/O Processor 10-157
VAX/VMS Resident 8086/8088/80186 Software Development Packages 22-383
VLSICE™-96 In-Circuit Emulator for the MCS®-96 21-323

CUSTOMER SUPPORT

CUSTOMER SUPPORT

Customer Support is Intel's complete support service that provides Intel customers with hardware support, software support, customer training, and consulting services. For more information contact your local sales offices.

After a customer purchases any system hardware or software product, service and support become major factors in determining whether that product will continue to meet a customer's expectations. Such support requires an international support organization and a breadth of programs to meet a variety of customer needs. As you might expect, Intel's customer support is quite extensive. It includes factory repair services and worldwide field service offices providing hardware repair services, software support services, customer training classes, and consulting services.

HARDWARE SUPPORT SERVICES

Intel is committed to providing an international service support package through a wide variety of service offerings available from Intel Hardware Support.

SOFTWARE SUPPORT SERVICES

Intel's software support consists of two levels of contracts. Standard support includes TIPS (Technical Information Phone Service), updates and subscription service (product-specific troubleshooting guides and COMMENTS Magazine). Basic support includes updates and the subscription service. Contracts are sold in environments which represent product groupings (i.e., iRMX environment).

CONSULTING SERVICES

Intel provides field systems engineering services for any phase of your development or support effort. You can use our systems engineers in a variety of ways ranging from assistance in using a new product, developing an application, personalizing training, and customizing or tailoring an Intel product to providing technical and management consulting. Systems Engineers are well versed in technical areas such as microcommunications, real-time applications, embedded microcontrollers, and network services. You know your application needs; we know our products. Working together we can help you get a successful product to market in the least possible time.

CUSTOMER TRAINING

Intel offers a wide range of instructional programs covering various aspects of system design and implementation. In just three to ten days a limited number of individuals learn more in a single workshop than in weeks of self-study. For optimum convenience, workshops are scheduled regularly at Training Centers worldwide or we can take our workshops to you for on-site instruction. Covering a wide variety of topics, Intel's major course categories include: architecture and assembly language, programming and operating systems, bitbus and LAN applications.

Preface

Preface

Computer systems can be characterized as being "reprogrammable" or "embedded". Reprogrammable systems are those that look and behave like computers to the ultimate user. An obvious example is the personal computer. These systems contain some form of mass storage device which stores a number of different computer programs that the user may call up and use as required. The input and output devices attached to these systems are there to communicate with the user. Embedded systems, as the name implies, are contained within a final product which may not look or feel like a computer to the end user. An example here is a computer which controls the ubiquitous office copier machine. These systems rarely contain mass storage; the programs are stored in ROM or EPROM devices. The input and output devices attached are not limited to communication with the user (e.g. the control panel); they also monitor and control mechanisms and processes within the device (e.g. the paper feed mechanism).

Embedded control applications can be broken into three broad categories; sequential control, closed loop control, and data control. Sequential control deals with the control and monitoring of a system as a sequence of events; activate the paper feed roller, wait for the paper feed indicator then activate the drum mechanism. Closed loop control involves closely monitoring the output of process or device and altering its inputs to achieve the desired output; if the feed roller motor is turning too slowly or is decelerating then increase the drive to the motor to compensate. These two categories involve fixed programs that interface directly with the outside world. The data structures involved are normally small and simple. The third category of embedded control (data control) still runs fixed programs but the interface to the outside word becomes more indirect and the data structures become larger and more complex. A high end copier might digitize an image and then use image processing techniques to enhance contrast and then scale and rotate the image to fit the paper being used.

Intel has consolidated four of its product families intended for embedded control applications into the Embedded Control Operation.

MCS®-48: Designed for general purpose 8-bit sequential control applications. Versions of these parts are available with program storage and up to 256 bytes of data storage onboard. The maximum program size is 4 Kbytes.

MCS®-51: Designed for advanced 8-bit sequential control applications. These parts are similar to the MCS®-48 family parts but operate from 2 to 5 times faster and include more on-board peripherals. A unique feature of the MCS®-51 family is a built in Boolean processor which performs calculations on Boolean (one bit) variables. The maximum program size is 64 Kbytes.

MCS®-96: Designed for advanced 16-bit closed loop control applications. These parts include a processor capable of high performance integer arithmetic and 232 bytes of general purpose registers which can be used for byte, word, or double-word operands. A separate subsystem manages timer functions. Versions are available with on-board A/D conversion and 8 Kbytes of on-board program memory. A unique feature of the newer (BH) members of the family is dynamic bus sizing which allows the parts to operate on both 8-bit and 16-bit busses. The maximum program size is 64 Kbytes.

80186/80188: These parts are highly integrated versions of the 8086 microprocessor intended for data control applications. They combine 15 to 20 of the most common 8086 system components onto one device. Included with the CPU is the clock generator, an interrupt controller, timers, DMA channels and chip select logic. The 80186 operates on a 16-bit bus and the 80188 operates on an 8-bit bus. Both parts operate on a 16-bit bus internally.

Part of the motivation for combining all of these products into one handbook was the realization that, although the categorization of embedded control applications into the three segments (sequential, closed loop, and data control) is useful for conceptualizing application requirements, real applications of these parts contain attributes from all three categories. Combining these product lines into one handbook will make it easier for customers involved with embedded applications to find the information they require.

The remainder of this chapter will give a very brief overview of each of the four products discussed. The remainder of this book is intended to provide detailed technical information on Intel's embedded control product line.

MCS®-48 MICROCONTROLLERS

Intel's MCS-48 family of 8-bit microcontrollers has become a world standard and has been in production for 10 years. They are available in several versions: with on board ROM, on board EPROM, or CPU only, to better fit specific application needs. MCS-48 products are now fabricated on advanced HMOS processes offering higher performance and reliability with less power consumption.

Features common to all members of this family are:

- 8-bit CPU with 1.36 microsecond instruction cycle
- 27 I/O lines
- 8-bit bit timer/counter
- 2 interrupts
- 4 Kbyte maximum program size
- 256 byte maximum on-board RAM size
- 256 byte maximum off-board RAM size

Intel has over ten years of experience in manufacturing both the EPROM and ROM versions of this chip. It provides a low cost solution to applications such as keyboards, low end printers, and electronic carburator control.

Table 1. MCS®-48 Microcontrollers

Device Name	ROMless Version	EPROM Version	ROM Bytes	RAM Bytes
8048AH	8035AHL	8748H	1K	64
8049AH	8039AHL	8749H	2K	128
8050AH	8040AHL	—	4K	256

MCS®-51 MICROCONTROLLERS

Intel's MCS-51 family is the industry standard for 8-bit high performance microcontrollers. The family architecture is optimized for sequential real-time control applications. They are available in several versions with on-board ROM, on-board EPROM, and CPU only to better fit specific application needs. MCS-51 products are available on advanced HMOS and CHMOS processes.

Features common to all members of this family are:

- 8-bit CPU optimized for control applications
- Extensive Boolean processing (single-bit logic) capabilities
- 32 bidirectional and individually addressable I/O lines
- Full-Duplex UART
- 5 source interrupt structure with 2 priority levels (6 sources on parts with 3 timer/counters)
- 64 Kbyte maximum program size
- 256 byte maximum on-board RAM size
- 64 Kbyte maximum off-board RAM size
- Power down/Idle modes for CHMOS parts

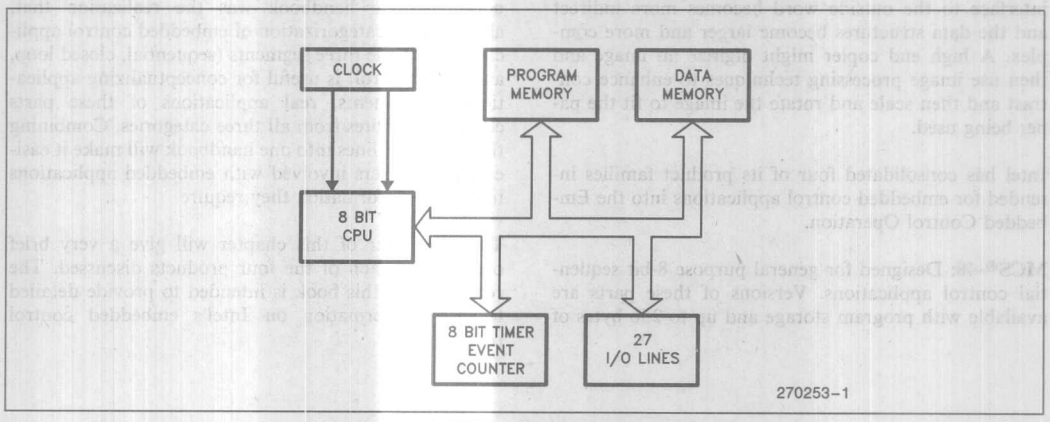


Figure 1. MCS®-48 Block Diagram

The MCS-51 family has found wide acceptance throughout a wide range of applications, ranging from those slightly more complex than a typical MCS-48 application through medical instrumentation and anti-skid braking modules for automobiles.

This year Intel added two new base products to the MCS-51 family, the 80C152 and the 80C51FA. The 80C152 contains, in addition to a UART, a Global Serial Channel capable of CSMA/CD and SDLC synchronous communication. It also has two DMA channels, the first member of the MCS-51 to have such capability. The 80C51FA contains, in addition to the standard timer counters, a programmable counter array (PCA) capable of measuring and generating pulse information on five I/O pins.

MSC®-51 FAMILY DEVELOPMENT TOOLS

ICE™-5100 emulators give design engineers full-speed, real-time, nonintrusive control over 8051 family system debugging at clock speeds up to 16 MHz. Each

Table 2. Advanced 8-Bit Microcontrollers

Device Name	ROMless Version	EPROM Version	ROM Bytes	RAM Bytes	16-Bit Timers	Circuit Type
8051	8031	(8751)	4K	128	2	HMOS
8051AH	8031AH	8751H	4K	128	2	HMOS
8052AH	8032AH	8752BH	8K	256	3	HMOS
80C51BH	80C31BH	87C51	4K	128	2	CHMOS
83C152	80C152		8K	256	2	CHMOS
83C51FA	80C51FA	87C51FA	8K	256	4	CHMOS

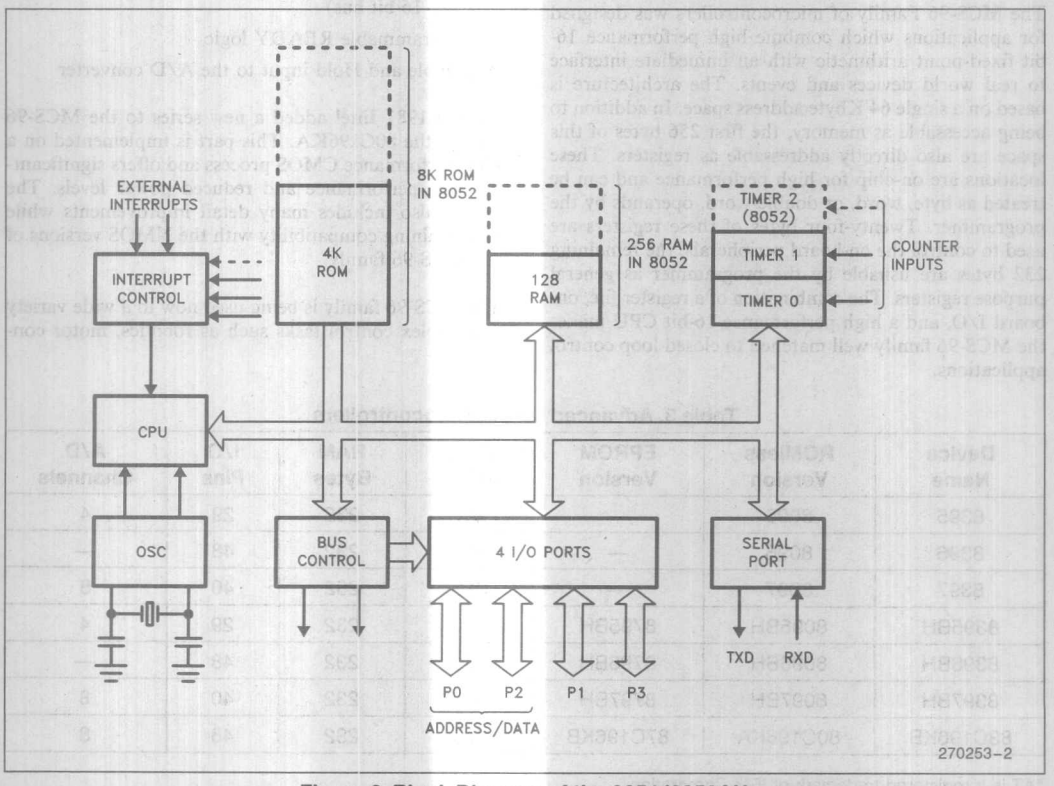


Figure 2. Block Diagram of the 8051/8052AH

emulator lets the user view and modify system activity at a symbolic, high-level language level, speeding and simplifying the development and debug phases of microcontroller system design. All of the ICE-5100 emulators can be hosted on IBM PC AT*s or compatibles, or Intellec® Series III/IV development systems. Three versions of ICE-5100 are available today. ICE-5100-252 supports HMOS and CHMOS versions of the following components: 8031, 8051, 8751, 8032, 8052, 8752, 80C31, 80C51, 87C51, 83C51FA, 80C51FA, and 87C51FA. ICE-5100/452 supports: 80C452, 83C452, 87C452. The ICE 5100/044 supports: 8344, 8044, 8744, and BITBUS™ components.

Available for the 51 family, ASM-51 and PL/M-51 both contain a relocation and linkage utility and are available for the IBM PC and Intel Microcomputer Development Systems running either iNDX or ISIS operating systems.

This complete, integrated design-in solution for the MCS-51 family of microcontrollers speeds product development, and improves design team productivity.

MCS®-96 MICROCONTROLLERS

The MCS-96 Family of microcontrollers was designed for applications which combine high performance 16-bit fixed-point arithmetic with an immediate interface to real world devices and events. The architecture is based on a single 64 Kbyte address space. In addition to being accessable as memory, the first 256 bytes of this space are also directly addressable as registers. These locations are on-chip for high performance and can be treated as byte, word, or double-word, operands by the programmer. Twenty-four bytes of these registers are used to control the on-board peripherals; the remaining 232 bytes are usuable by the programmer as general purpose registers. The combination of a register file, on-board I/O, and a high performance 16-bit CPU makes the MCS-96 family well matched to closed loop control applications.

Features common to all members of this family are:

- 16-bit timer (Timer1)
- 16-bit counter (Timer2)
- Full-Duplex UART with independent baud-rate generator
- Watchdog timer
- 8-bit resolution Pulse Width Modulator (PWM)
- 48 I/O lines (33 for 48-pin parts)
- HSIO unit

The HSIO (High Speed Input/Output) unit is an independent timer subsystem which manages Timer1 and Timer2 for the programmer. Events (e.g. setting an I/O pin) can be scheduled to occur automatically when either of the two timers reaches a preset value. External events can be recorded in a FIFO along with the value of Timer1 when the event occurred. The HSIO is connected to eight pins on the 8096 and can generate and monitor events with a 2 microsecond resolution (12 MHz crystal).

In addition to an EPROM version, the 8X9XBH offer several improvements over the non BH parts:

- Dynamic selection of 8-/16-bit bus operation (versus fixed 16-bit bus)
- Programmable READY logic
- Sample and Hold input to the A/D converter

During 1987 Intel added a new series to the MCS-96 family, the 80C196KA. This part is implemented on a high performance CMOS process and offers significantly more performance and reduced power levels. The design also includes many detail improvements while still retaining compatibility with the NMOS versions of the MCS-96 family.

The MCS-96 family is being used now in a wide variety of complex control tasks such as robotics, motor con-

Table 3. Advanced 16-Bit Microcontrollers

Device Name	ROMless Version	EPROM Version	ROM Bytes	RAM Bytes	I/O Pins	A/D Channels
8395	8095	—	8K	232	29	4
8396	8096	—	8K	232	48	—
8397	8097	—	8K	232	40	8
8395BH	8095BH	8795BH	8K	232	29	4
8396BH	8096BH	8796BH	8K	232	48	—
8397BH	8097BH	8797BH	8K	232	40	8
83C196KB	80C196KA	87C196KB	8K	232	48	8

*AT is a registered trademark of IBM Corporation.

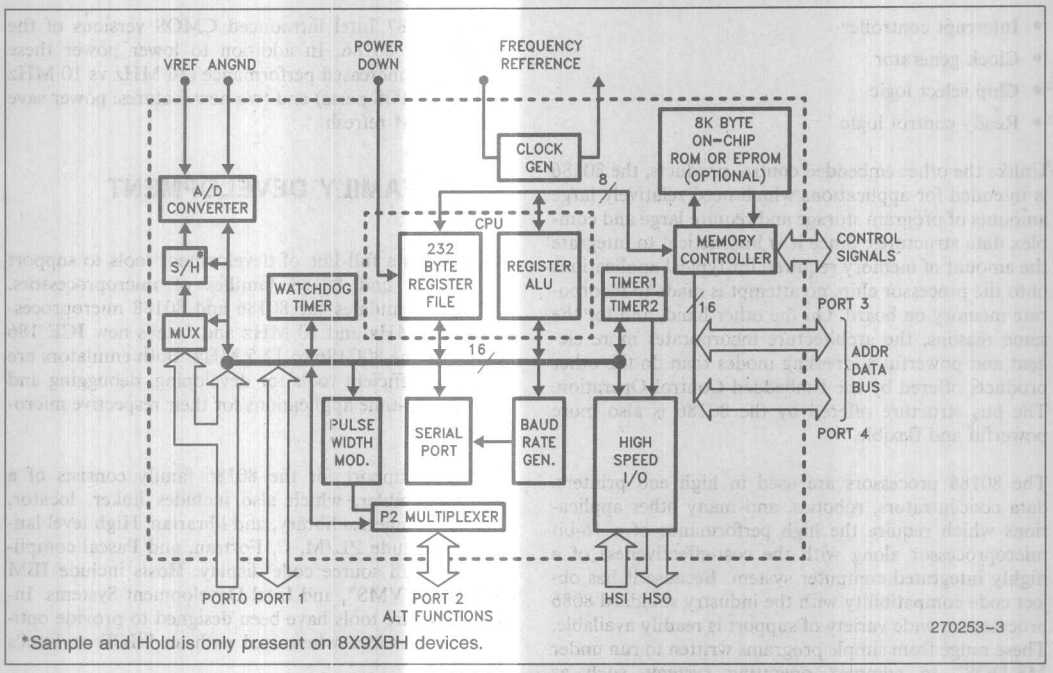


Figure 3. MCS®-96 Block Diagram

trol, hard disk mechanisms, printers, modems, automotive engine control and high performance anti-skid braking applications.

MCS®-96 FAMILY DEVELOPMENT TOOLS

Intel offers a variety of development tools to support the MCS-96 family of microcontrollers. The iSBE-96, a low-cost single board emulator, emulates the 8X9X microcontrollers. VLSiCE-96 provides complete in-circuit emulation of the 80C196KC microcontroller. The ICE™-196PC In-Circuit emulator provides real-time, transparent emulation of the 80C196KA microcontroller. All three emulators are efficient and versatile tools for developing, debugging and testing real-time MCS-96 applications.

Software for the MCS-96 family includes macroassembler (ASM-96) which is available along with PL/M-96 and C-96 compilers. Each software package includes relocation/linkage utility, library creation utility, and FPAL-96, a 32-bit floating-point utility. Software runs on IBM PC and Intellec Series III/IV.

80186 HIGH INTEGRATION MICROPROCESSOR

The 80186 family of microprocessors was designed to bring the 8086 architecture to embedded control applications. Six of the most often used functions of an 8086 system have been integrated onto a single chip. Along with higher integration, the 80186 offers higher performance than the standard 8086 via detailed improvements to the design. The 80186 family consists of two processors; the 80186 and the 80188. These processors have identical capabilities except that the 80186 operates on a 16-bit bus while the 80188 operates on an 8-bit bus. The 80186 offers higher system performance with its 16-bit bus; the 80188 offers lower system cost with its 8-bit bus. Both processors operate internally with a 16-bit bus and generates a 20-bit address to give a total address space of 1 Megabyte.

Although these parts are object code compatible with the 8086, most of the instructions complete in fewer clock cycles because of enhanced CPU and bus control logic. In addition, many new instructions have been included to simplify assembly language programs and to enhance operation with high level languages. These new instructions have been included in the 80286 and 80386 high performance microprocessors.

Besides the processor itself, the major features of the 80186 and 80188 include:

- 2 Channel DMA (Direct Memory Access) unit
- 3 16-bit timer/counters

- Interrupt controller
- Clock generator
- Chip select logic
- Ready control logic

Unlike the other embedded control products, the 80186 is intended for applications which need relatively large amounts of program storage and require large and complex data structures. Since it is impractical to integrate the amount of memory required for typical applications onto the processor chip, no attempt is made to incorporate memory on board. On the other hand, and for the same reasons, the architecture incorporates more elegent and powerful addressing modes than do the other products offered by the Embedded Control Operation. The bus structure offered by the 80186 is also more powerful and flexible.

The 80186 processors are used in high end printers, data concentrators, robotics, and many other applications which require the high performance of a 16-bit microprocessor along with the cost-effectiveness of a highly integrated computer system. Because it has object code compatibility with the industry standard 8086 processor, a wide variety of support is readily available. These range from simple programs written to run under MS-DOS* to complex operating systems such as iRMX®-86 and XENIX*.

During 1987 Intel introduced CMOS versions of the 80188 and 80186. In addition to lower power these parts offer increased performance (16 MHz vs 10 MHz for the NMOS parts) and two new features: power save and DRAM refresh.

80186 FAMILY DEVELOPMENT TOOLS

Intel offers a full line of development tools to support the 80186 and 80188 families of microprocessors. I²ICE™ emulates the 80186 and 80188 microprocessors at 8 MHz and 10 MHz and Intel's new ICE 186 emulates the 80C186 to 12.5 MHz. Both emulators are versatile, efficient tools for developing, debugging and testing real-time applications for their respective microprocessors.

Software support for the 80186 family consists of a macroassembler, which also includes linker, locator, mapper, numerics library, and librarian. High level languages include PL/M, C, Fortran, and Pascal compilers with full source code display. Hosts include IBM PC, VAX/VMS*, and Intel Development Systems. Intel's software tools have been designed to provide optimal performance when used with an I²ICE or Intel's ICE-186.

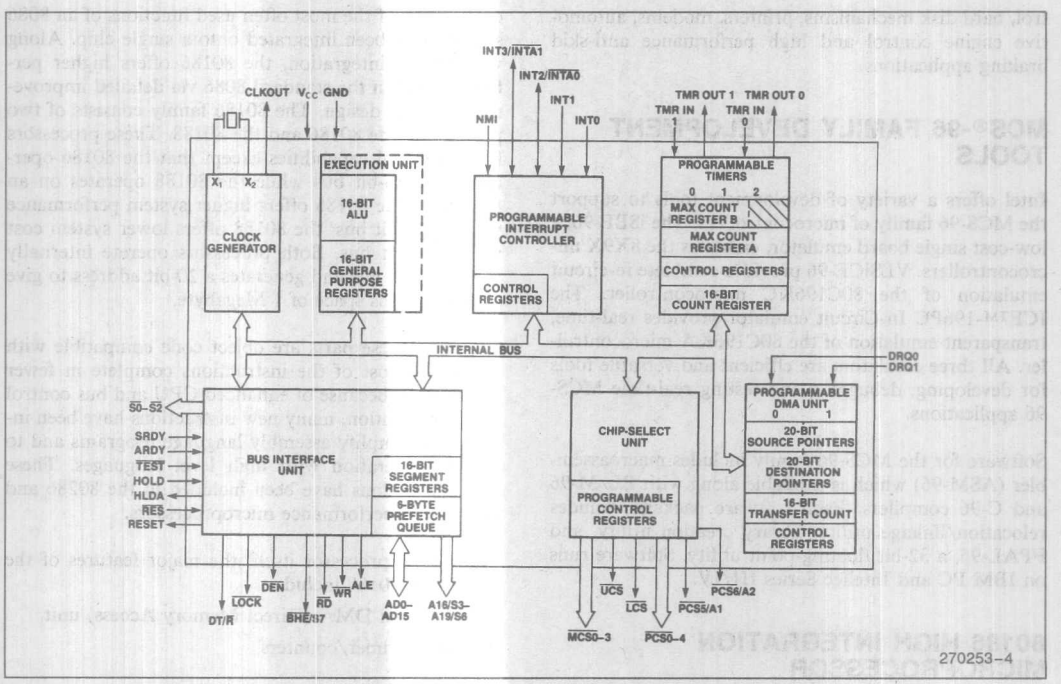


Figure 4. 80186/80188 Block Diagram—A CPU Board On a Single Silicon Chip

*VAX/VMS is a registered trademark of Digital Equipment Corporation.
*MS-DOS and XENIX are trademarks of Microsoft Corporation.

MCS®-48 Single Component System

1

THE SINGLE COMPONENT MCS®-48 SYSTEM

1.0 INTRODUCTION

Sections 2 through 5 describe in detail the functional characteristics of the 8748H and 8749H EPROM, 8048AH/8049AH/8050AH ROM, and 8035AHL/8039AHL/8040-AHL CPU only single component microcomputers. Unless otherwise noted, details within these sections apply to all versions. This chapter is limited to those functions useful in single-chip implementations of the MCS®-48. The Chapter on the Expanded MCS®-48 System discusses functions which allow expansion of program memory, data memory, and input output capability.

2.0 ARCHITECTURE

The following sections break the MCS-48 Family into functional blocks and describe each in detail. The following description will use the 8048AH as the representative product for the family. See Figure 1.

2.1 Arithmetic Section

The arithmetic section of the processor contains the basic data manipulation functions of the 8048AH and can be divided into the following blocks:

- Arithmetic Logic Unit (ALU)
- Accumulator
- Carry Flag
- Instruction Decoder

In a typical operation data stored in the accumulator is combined in the ALU with data from another source on the internal bus (such as a register or I/o port) and the result is stored in the accumulator or another register.

The following is more detailed description of the function of each block.

INSTRUCTION DECODER

The operation code (op code) portion of each program instruction is stored in the Instruction Decoder and converted to outputs which control the function of each of the blocks of the Arithmetic Section. These lines control the source of data and the destination register as well as the function performed in the ALU.

ARITHMETIC LOGIC UNIT

The ALU accepts 8-bit data words from one or two sources and generates an 8-bit result under control of the Instruction Decoder. The ALU can perform the following functions:

- Add With or Without Carry
- AND, OR, Exclusive OR
- Increment/Decrement
- Bit Complement
- Rotate Left, Right
- Swap Nibbles
- BCD Decimal Adjust

If the operation performed by the ALU results in a value represented by more than 8 bits (overflow of most significant bit), a Carry Flag is set in the Program Status Word.

ACCUMULATOR

The accumulator is the single most important data register in the processor, being one of the sources of input to the ALU and often the destination of the result of operations performed in the ALU. Data to and from I/O ports and memory also normally passes through the accumulator.

2.2 Program Memory

Resident program memory consists of 1024, 2048, or 4096 words eight bits wide which are addressed by the program counter. In the 8748H and the 8749H this memory is user programmable and erasable EPROM; in the 8048AH/8049AH/8050AH the memory is ROM which is mask programmable at the factory. The 8035AHL/8039AHL/8040AHL has no internal program memory and is used with external memory devices. Program code is completely interchangeable among the various versions. To access the upper 2K of program memory in the 8050AH, and other MCS-48 devices, a select memory bank and a JUMP or CALL instruction must be executed to cross the 2K boundary.

There are three locations in Program Memory of special importance as shown in Figure 2.

LOCATION 0
Activating the Reset line of the processor causes the first instruction to be fetched from location 0.

LOCATION 3
Activating the Interrupt input line of the processor (if interrupt is enabled) causes a jump to subroutine at location 3.

LOCATION 7
A timer/counter interrupt resulting from timer counter overflow (if enabled) causes a jump to subroutine at location 7.

Therefore, the first instruction to be executed after initialization is stored in location 0, the first word of an external interrupt service subroutine is stored in location 3, and the first word of a timer/counter service routines

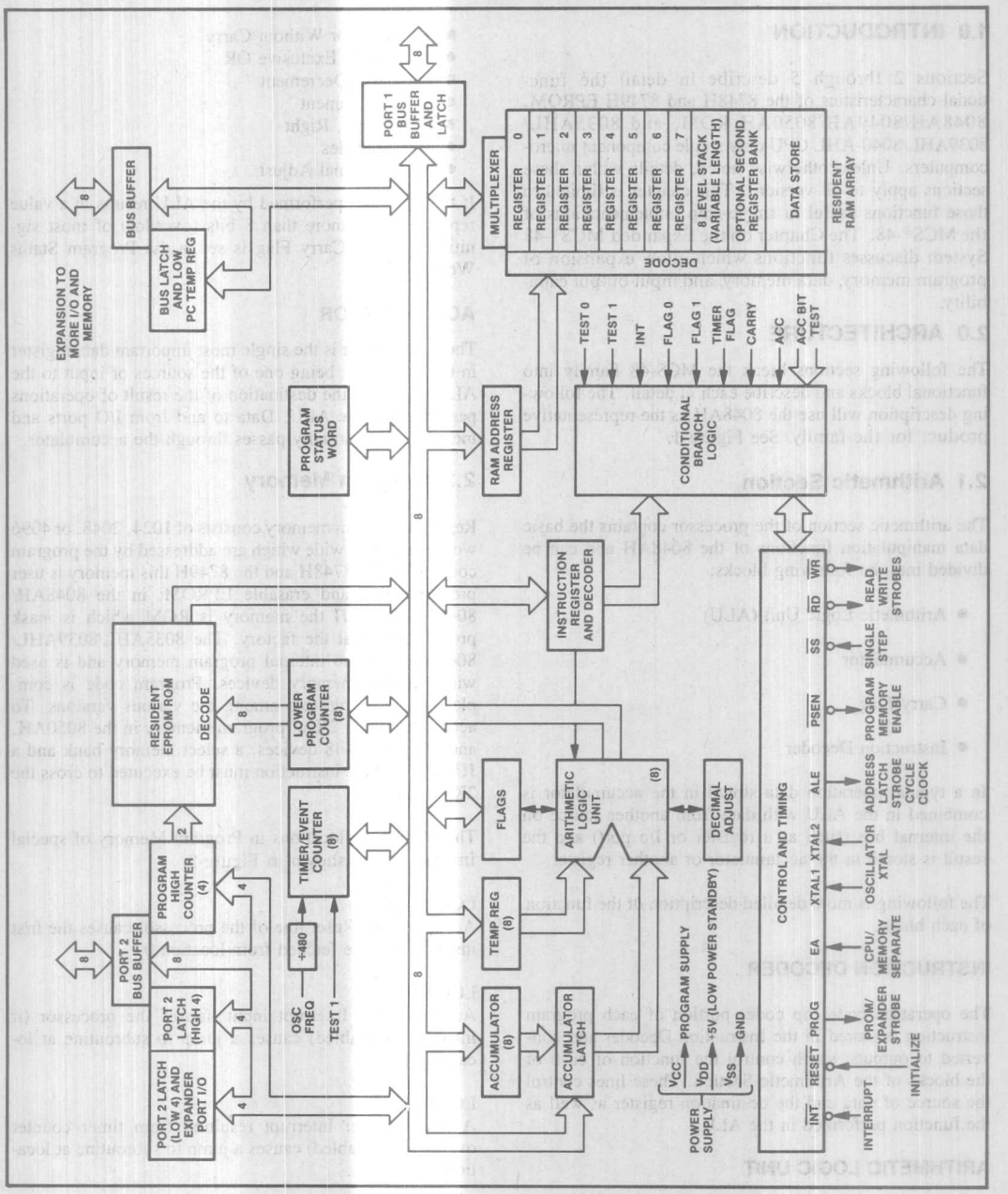


Figure 1. 8748H/8048H/8749AH/8050AH Block Diagram

is stored in location 7. Program memory can be used to store constants as well as program instructions. Instructions such as MOVP and MOVP3 allow easy access to data "lookup" tables.

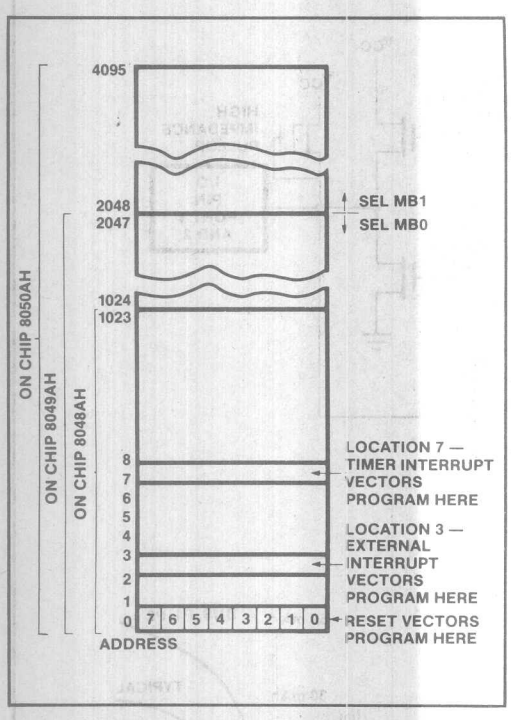


Figure 2. Program Memory Map

2.3 Data Memory

Resident data memory is organized as 64, 128, or 256 by 8-bits wide in the 8048AH, 8049AH and 8050AH. All locations are indirectly addressable through either of two RAM Pointer Registers which reside at address 0 and 1 of the register array. In addition, as shown in Figure 3, the first 8 locations (0–7) of the array are designated as working registers and are directly addressable by several instructions. Since these registers are more easily addressed, they are usually used to store frequently accessed intermediate results. The DJNZ instruction makes very efficient use of the working registers as program loop counters by allowing the programmer to decrement and test the register in a single instruction.

By executing a Register Bank Switch instruction (SEL RB) RAM locations 24–31 are designated as the working registers in place of locations 0–7 and are then directly addressable. This second bank of working registers may be used as an extension of the first bank or reserved for use during interrupt service subroutines allowing the registers of Bank 0 used in the main program to be instantly "saved" by a Bank Switch. Note that if this second bank is not used, locations 24–31 are still addressable as general purpose RAM. Since the two RAM pointer Registers R0 and R1 are a part of the working register array, bank switching effectively creates two more pointer registers (R0/and R1/) which can be used with R0 and R1 to easily access up to four separate working areas in RAM at one time. RAM locations (8–23) also serve a dual role in that they contain the program counter stack as explained in Section 2.6. These locations are addressed by the Stack Pointer during subroutine calls as well as by RAM Pointer Registers R0 and R1. If the level of subroutine nesting is less than 8, all stack registers are not required and can be used as general purpose RAM locations. Each level of subroutine nesting not used provides the user with two additional RAM locations.

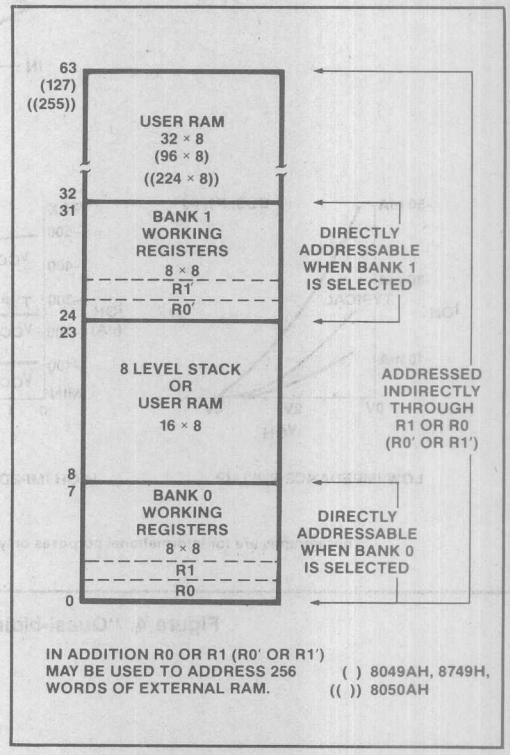


Figure 3. Data Memory Map

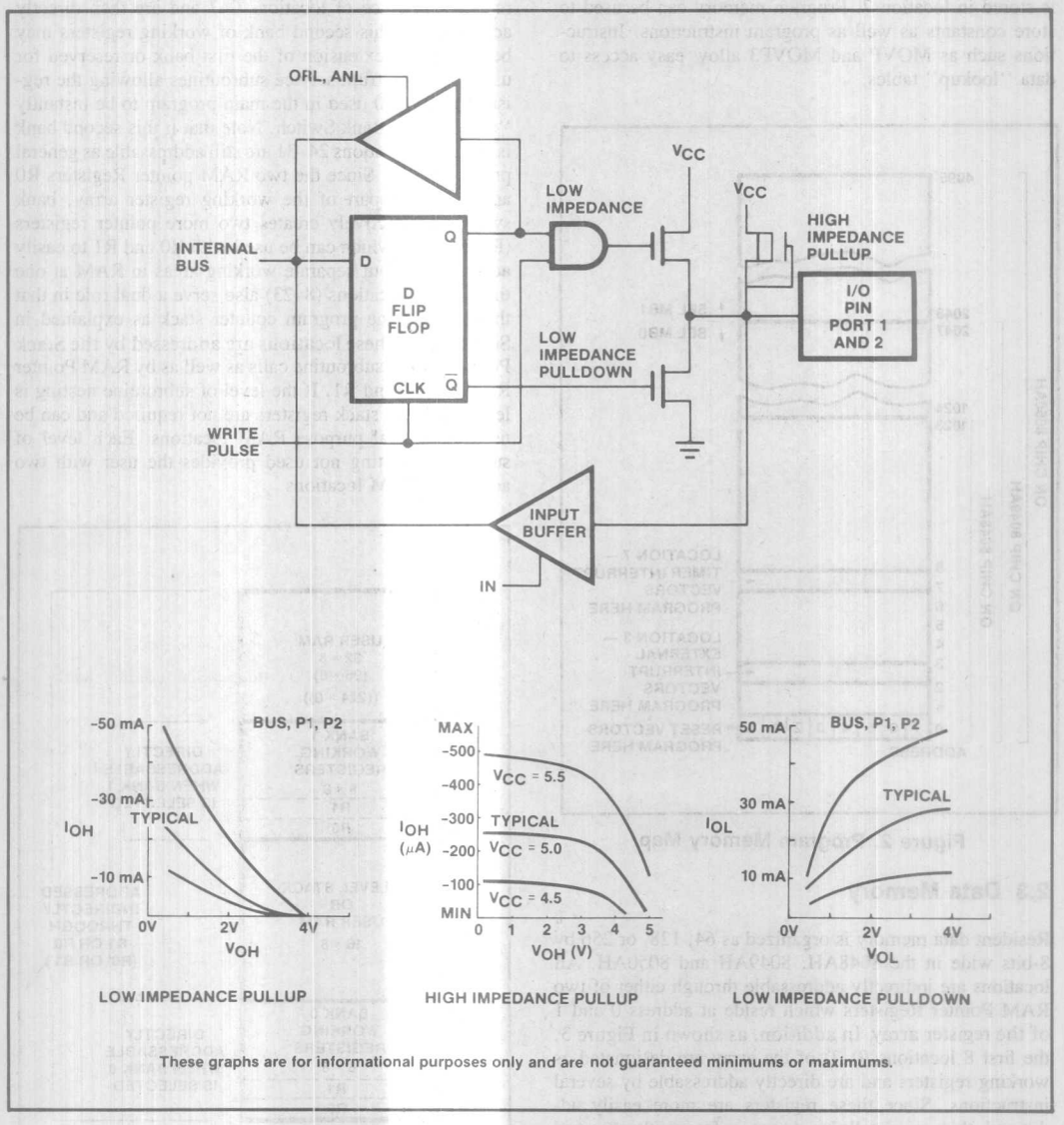


Figure 4. "Quasi-bidirectional" Port Structure

2.4 Input/Output

The 8048AH has 27 lines which can be used for input or output functions. These lines are grouped as 3 ports of 8 lines each which serve as either inputs, outputs or bidirectional ports and 3 "test" inputs which can alter program sequences when tested by conditional jump instructions.

PORTS 1 AND 2

Ports 1 and 2 are each 8 bits wide and have identical characteristics. Data written to these ports is statically latched and remains unchanged until rewritten. As input ports these lines are non-latching, i.e., inputs must be present until read by an input instruction. Inputs are fully TTL compatible and outputs will drive one standard TTL load.

The lines of ports 1 and 2 are called quasi-bidirectional because of a special output circuit structure which allows each line to serve as an input, and output, or both even though outputs are statically latched. Figure 4 shows the circuit configuration in detail. Each line is continuously pulled up to V_{CC} through a resistive device of relatively high impedance.

This pullup is sufficient to provide the source current for a TTL high level yet can be pulled low by a standard TTL gate thus allowing the same pin to be used for both input and output. To provide fast switching times in a "0" to "1" transition a relatively low impedance device is switched in momentarily ($\approx$ 1/5 of a machine cycle) whenever a "1" is written to the line. When a "0" is written to the line a low impedance device overcomes the light pullup and provides TTL current sinking capability. Since the pulldown transistor is a low impedance device a "1" must first be written to any line which is to be used as an input. Reset initializes all lines to the high impedance "1" state.

It is important to note that the ORL and the ANL are read/write operations. When executed, the μC "reads" the port, modifies the data according to the instruction, then "writes" the data back to the port. The "writing" (essentially an OUTL instruction) enables the low impedance pull-up momentarily again even if the data was unchanged from a "1." This specifically applies to configurations that have inputs and outputs mixed together on the same port. See also section 8 in the Expanded MCS-48 System chapter.

BUS

Bus is also an 8-bit port which is a true bidirectional port with associated input and output strobes. If the bidirectional feature is not needed, Bus can serve as either a statically latched output port or non-latching input port. Input and output lines on this port cannot be mixed however.

As a static port, data is written and latched using the OUTL instruction and inputted using the INS instruction. The INS and OUTL instructions generate pulses on the corresponding $\overline{RD}$ and $\overline{WR}$ output strobe lines; however, in the static port mode they are generally not used. As a bidirectional port the MOVX instructions are used to read and write the port. A write to the port generates a pulse on the $\overline{WR}$ output line and output data is valid at the trailing edge of $\overline{WR}$. A read of the port generates a pulse on the $\overline{RD}$ output line and input data must be valid at the trailing edge of $\overline{RD}$. When not being written or read, the BUS lines are in a high impedance state. See also sections 7 and 8 in the Expanded MCS-48 System chapter.

2.5 Test and INT Inputs

Three pins serve as inputs and are testable with the conditional jump instruction. These are T0, T1, and $\overline{INT}$. These pins allow inputs to cause program branches without the necessity to load an input port into the accumulator. The T0, T1, and $\overline{INT}$ pins have other possible functions as well. See the pin description in Section 3.

2.6 Program Counter and Stack

The Program Counter is an independent counter while the Program Counter Stack is implemented suing pairs of registers in the Data Memory Array. Only 10, 11, or 12 bits of the Program Counter are used to address the 1024, 2048, or 4096 words of on-board program memory of the 8048AH, 8049AH, or 8050AH, while the most significant bits can be used for external Program Memory fetches. See Figure 5. The Program Counter is initialized to zero by activating the Reset line.

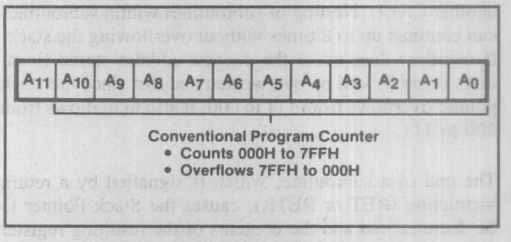


Figure 5. Program Counter

An interrupt or CALL to a subroutine causes the contents of the program counter to be stored in one of the 8 register pairs of the Program Counter Stack as shown in Figure 6. The pair to be used is determined by a 3-bit Stack Pointer which is part of the Program Status Word (PSW).

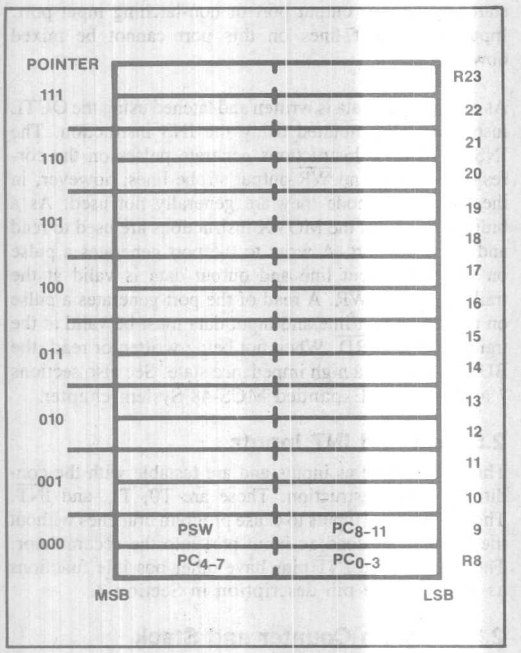


Figure 6. Program Counter Stack

Data RAM locations 8-23 are available as stack registers and are used to store the Program Counter and 4 bits of PSW as shown in Figure 6. The Stack Pointer .when initialized to 000 points to RAM locations 8 and 9. The first subroutine jump or interrupt results in the program counter contents being transferred to locations 8 and 9 of the RAM array. The stack pointer is then incremented by one to point to locations 10 and 11 in anticipation of another CALL. Nesting of subroutines wihtin subroutines can continue up to 8 times without overflowing the stack. If overflow does occur the deepest address stored (locations 8 and 9) will be overwritten and lost since the stack pointer overflows from 111 to 000. It also underflows from 000 to 111.

The end of a subroutine, which is signalled by a return instruction (RET or RETR), causes the Stack Pointer to be decremented and the contents of the resulting register pair to be transferred to the Program Counter.

2.7 Program Status Word

An 8-bit status word which can be loaded to and from the accumulator exists called the Program Status Word (PSW). Figure 7 shows the information available in the word. The Program Status Word is actually a collection of flip-flops throughout the machine which can be read or written as a whole. The ability to write to PSW allows for easy restoration of machine status after a power down sequence.

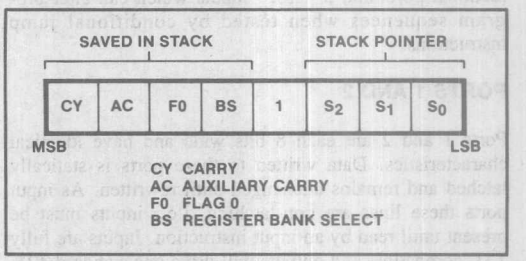


Figure 7. Program Status Word (PSW)

The upper four bits of PSW are stored in the Program Counter Stack with every call to subroutine or interrupt vector and are optionally restored upon return with the RETR instruction. The RET return instruction does not update PSW.

The PSW bit definitions are as follows:

Bits 0–2: Stack Pointer bits (S_0, S_1, S_2)

Bit 3: Not used (''1'' level when read)

Bit 4: Working Register Bank Switch Bit (BS)
0 = Bank 0
1 = Bank 1

Bit 5: Flag 0 bit (F0) user controlled flag which can be complemented or cleared, and tested with the conditional jump instruction JF0.

Bit 6: Auxiliary Carry (AC) carry bit generated by an ADD instruction and used by the decimal adjust instruction DA A.

Bit 7: Carry (CY) carry flag which indicates that the previous operation has resulted in overflow of the accumulator.

2.8 Conditional Branch Logic

The conditional branch logic within the processsor enables several conditions internal and external to the processor to be tested by the users program. By using the conditional jump instruction the conditions that are listed in Table 1 can effect a change in the sequence of the program execution.

Table 1

Device Testable	Jump Conditions (Jump On)	
Accumulator	All zeros	not all zeros
Accumulator Bit	—	1
Carry Flag	0	1
User Flags (F0, F1)	—	1
Timer Overflow Flag	—	1
Test Inputs (T0, T1)	0	1
Interrupt Input ($\overline{INT}$)	0	—

2.9 Interrupt

An interrupt sequence is initiated by applying a low "0" level input to the INT pin. Interrupt is level triggered and active low to allow "WIRE ORing" of several interrupt sources at the input pin. Figure 8 shows the interrupt logic of the 8048AH. The Interrupt line is sampled every instruction cycle and when detected causes a "call to subroutine" at location 3 in program memory as soon as all cycles of the current instruction are complete. On 2-cycle instructions the interrupt line is sampled on the 2nd cycle only. $\overline{INT}$ must be held low for at least 3 machine cycles to ensure proper interrupt operations. As in any CALL to subroutine, the Program Counter and Program Status word are saved in the stack. For a description of this operation see the previous section, Program Counter and Stack. Program Memory location 3 usually contains an unconditional jump to an interrupt service subroutine elsewhere in program memory. The end of an interrupt service subroutine is signalled by the execution of a Return and Restore Status instruction RETR. The interrupt system is single level in that once an interrupt is detected all further interrupt requests are ignored until execution of an RETR reenables the interrupt input logic. This occurs at the beginning of the second cycle of the RETR instruction. This sequence holds true also for an internal interrupt generated by timer overflow. If an internal timer/counter generated interrupt and an external interrupt are detected at the same time, the external source will be recognized. See the following Timer/Counter section for a description of timer interrupt. If needed, a second external interrupt can be created by enabling the timer/counter interrupt, loading FFH in the Counter (ones less than terminal count), and enabling the event counter mode. A "1" to "0" transition on the T1 input will then cause an interrupt vector to location 7.

INTERRUPT TIMING

The interrupt input may be enabled or disabled under Program Control using the EN I and DIS I instructions. Interrupts are disabled by Reset and remain so until enabled by the users program. An interrupt request must be removed before the RETR instruction is executed upon return from the service routine otherwise the processor will re-enter the service routine immediately. Many peripheral devices prevent this situation by resetting their interrupt request line whenever the processor accesses (Reads or Writes) the peripherals data buffer register. If the interrupting device does not require access by the processor, one output line of the 8048AH may be designated as an "interrupt acknowledge" which is activated by the service subroutine to reset the interrupt request. The $\overline{INT}$ pin may also be tested using the conditional jump instruction JNI. This instruction may be used to detect the presence of a pending interrupt before interrupts are enabled. If interrupt is left disabled, $\overline{INT}$ may be used as another test input like T0 and T1.

2.10 Timer/Counter

The 8048AH contains a counter to aid the user in counting external events and generating accurate time delays without placing a burden on the processor for these functions. In both modes the counter operation is the same, the only difference being the source of the input to the counter. The timer/event counter is shown in Figure 9.

COUNTER

The 8-bit binary counter is presettable and readable with two MOV instructions which transfer the contents of the accumulator to the counter and vice versa. The counter content may be affected by Reset and should be initialized by software. The counter is stopped by a Reset or STOP TCNT instruction and remains stopped until started as a timer by a START T instruction or as an event counter by a START CNT instruction. Once started the counter will increment to this maximum count (FF) and overflow to zero continuing its count until stopped by a STOP TCNT instruction or Reset.

The increment from maximum count to zero (overflow) results in the setting of an overflow flag flip-flop and in the generation of an interrupt request. The state of the overflow flag is testable with the conditional jump instruction JTF. The flag is reset by executing a JTF or by Reset. The interrupt request is stored in a latch and then ORed with the external interrupt input INT. The timer interrupt may be enabled or disabled independently of external interrupt by the EN TCNT1 and DIS TCNT1 instructions. If enabled, the counter overflow will cause a subroutine call to location 7 where the timer or counter service routine may be stored.

If timer and external interrupts occur simultaneously, the external source will be recognized and the Call will be to

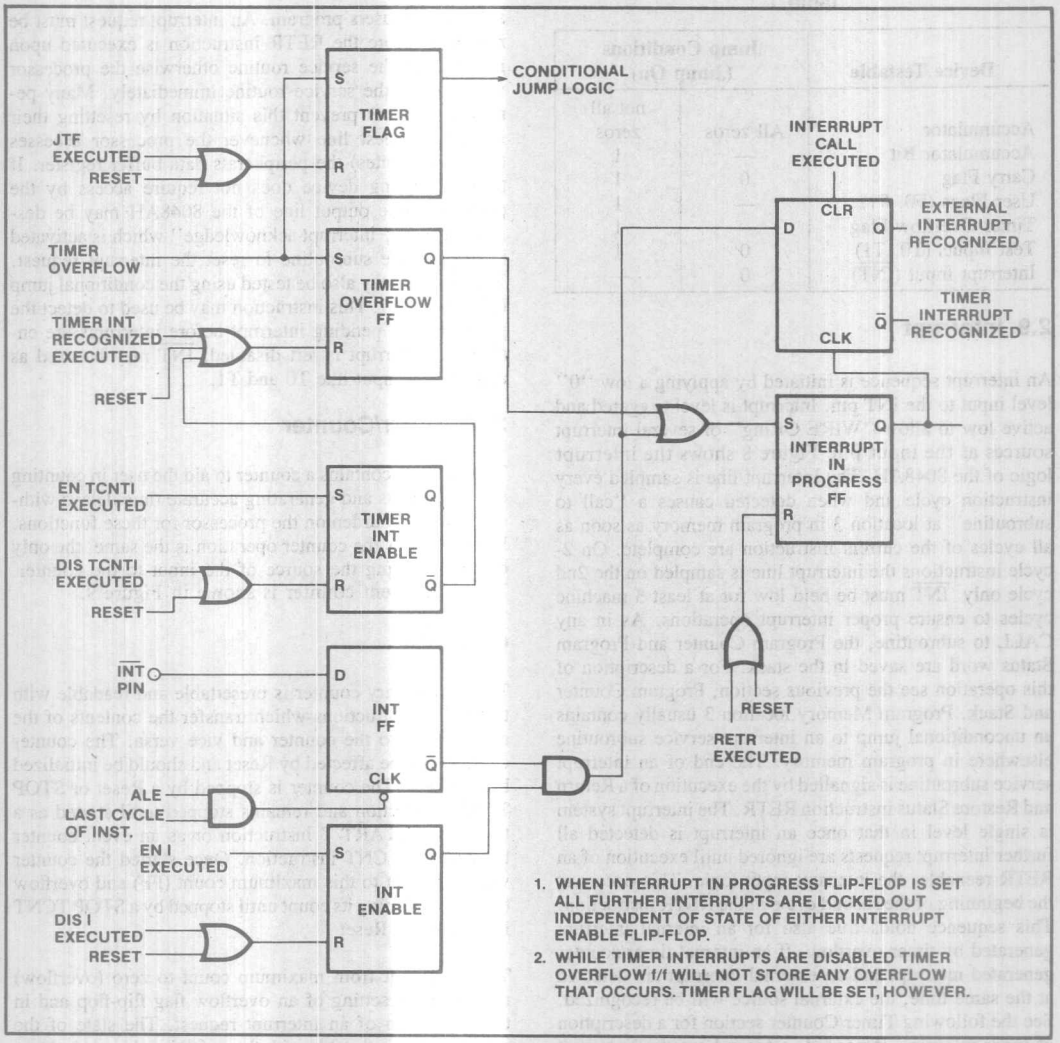


Figure 8. Interrupt Logic

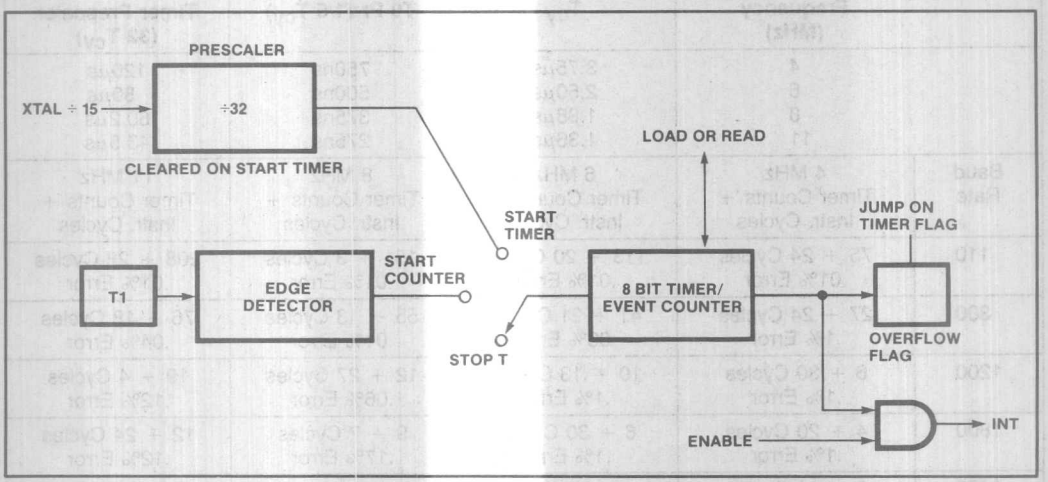


Figure 9. Timer/Event Counter

location 3. Since the timer interrupt is latched it will remain pending until the external device is serviced and immediately be recognized upon return from the service routine. The pending timer interrupt is reset by the Call to location 7 or may be removed by executing a DIS TCNT1 instruction.

AS AN EVENT COUNTER

Execution of a START CNT instruction connects the T1 input pin to the counter input and enables the counter. The T1 input is sampled at the beginning of state 3 or in later MCS-48 devices in state time 4. Subsequent high to low transitions on T1 will cause the counter to increment. T1 must be held low for at least 1 machine cycle to insure it won't be missed. The maximum rate at which the counter may be incremented is once per three instruction cycles (every 5.7 μsec when using an 8 MHz crystal) — there is no minimum frequency. T1 input must remain high for at least 1/5 machine cycle after each transition.

AS A TIMER

Eexcution of a START T instruction connects an internal clock to the counter input and enables the counter. The internal clock is derived bypassing the basic machine cycle clock through a ÷32 prescaler. The prescaler is reset during the START T instruction. The resulting clock increments the counter every 32 machine cycles. Various delays from 1 to 256 counts can be obtained by presetting the counter and detecting overflow. Times longer than 256 counts may be achieved by accumulating multiple overflows in a register under software control. For time resolution less than 1 count an external clock can be applied to the T1 input and the counter operated in the event counter mode. ALE divided by 3 or more can serve as this external clock. Very small delays or "fine tuning" of larger delays can be easily accomplished by software delay loops.

Often a serial link is desirable in an MCS-48 family member. Table 2 lists the timer counts and cycles needed for a specific baud rate given a crystal frequency.

2.11 Clock and Timing Circuits

Timing generation for the 8048AH is completely selfcontained with the exeception of a frequency reference which can be XTAL, ceramic resonator, or external clock source. The Clock and Timing circuitry can be divided into the following functional blocks.

OSCILLATOR

The on-board oscillator is a high gain parallel resonant circuit with a frequency range of 1 to 11 MHz. The X1 external pin is the input to the amplifier stage while X2 is the output. A crystal or ceramic resonator connected between X1 and X2 provides the feedback and phase shift required for oscillation. If an accurate frequency reference is not required, ceramic resonator may be used in place of the crystal.

For accurate clocking, a crystal should be used. An externally generated clock may also be applied to X1–X2 as the frequency source. See the data sheet for more information.

Table 2. Baud Rate Generation

	Frequency (MHz)	T_{cy}	T0 Prr(1/5 T_{cy})	Timer Prescaler (32 T_{cy})
	4	3.75μs	750ns	120μs
	6	2.50μs	500ns	80μs
	8	1.88μs	375ns	60.2μs
	11	1.36μs	275ns	43.5μs
Baud Rate	4 MHz Timer Counts + Instr. Cycles	6 MHz Timer Counts + Instr. Cycles	8 MHz Timer Counts + Instr. Cycles	11 MHz Timer Counts + Instr. Cycles
110	75 + 24 Cycles .01% Error	113 + 20 Cycles .01% Error	151 + 3 Cycles .01% Error	208 + 28 Cycles .01% Error
300	27 + 24 Cycles .1% Error	41 + 21 Cycles .03% Error	55 + 13 Cycles .01% Error	76 + 18 Cycles .04% Error
1200	6 + 30 Cycles .1% Error	10 + 13 Cycles .1% Error	12 + 27 Cycles .06% Error	19 + 4 Cycles .12% Error
1800	4 + 20 Cycles .1% Error	6 + 30 Cycles .1% Error	9 + 7 Cycles .17% Error	12 + 24 Cycles .12% Error
2400	3 + 15 Cycles .1% Error	5 + 6 Cycles .4% Error	6 + 24 Cycles .29% Error	9 + 18 Cycles .12% Error
4800	1 + 23 Cycles 1.0% Error	2 + 19 Cycles .4% Error	3 + 14 Cycles .74% Error	4 + 25 Cycles .12% Error

STATE COUNTER

The output of the oscillator is divided by 3 in the State Counter to create a clock which defines the state times of the machine (CLK). CLK can be made available on the external pin T0 by executing an ENTO CLK instruction. The output of CLK on T0 is disabled by Reset of the processor.

CYCLE COUNTER

CLK is then divided by 5 in the Cycle Counter to provide a clock which defines a machine cycle consisting of 5 machine states as shown in Figure 10. Figure 11 shows the different internal operations as divided into the machine states. This clock is called Address Latch Enable (ALE) because of its function in MCS-48 systems with external memory. It is provided continuously on the ALE output pin.

2.12 Reset

The reset input provides a means for initialization for the processor. This Schmitt-trigger input has an internal pull-up device which in combination with an external 1 μ fd capacitor provides an internal reset pulse of sufficient length to guarantee all circuitry is reset, as shown in Figure 12. If the reset pulse is generated externally the RESET pin must be held low for at least 10 milliseconds after the power supply is within tolerance.Only 5 machine cycles (6.8 μs @ 11 MHz) are required if power is already on and the oscillator has stabilized. ALE and PSEN (if EA = 1) are active while in Reset.

Reset performs the following functions:

1) Sets program counter to zero.
2) Sets stack pointer to zero.
3) Selects register bank 0.
4) Selects memory bank 0.
5) Sets BUS to high impedance state (except when EA = 5V).
6) Sets Ports 1 and 2 to input mode.
7) Disables interrupts (timer and external).
8) Stops timer.
9) Clears timer flag.
10) Clears F0 and F1.
11) Disables clock output from T0.

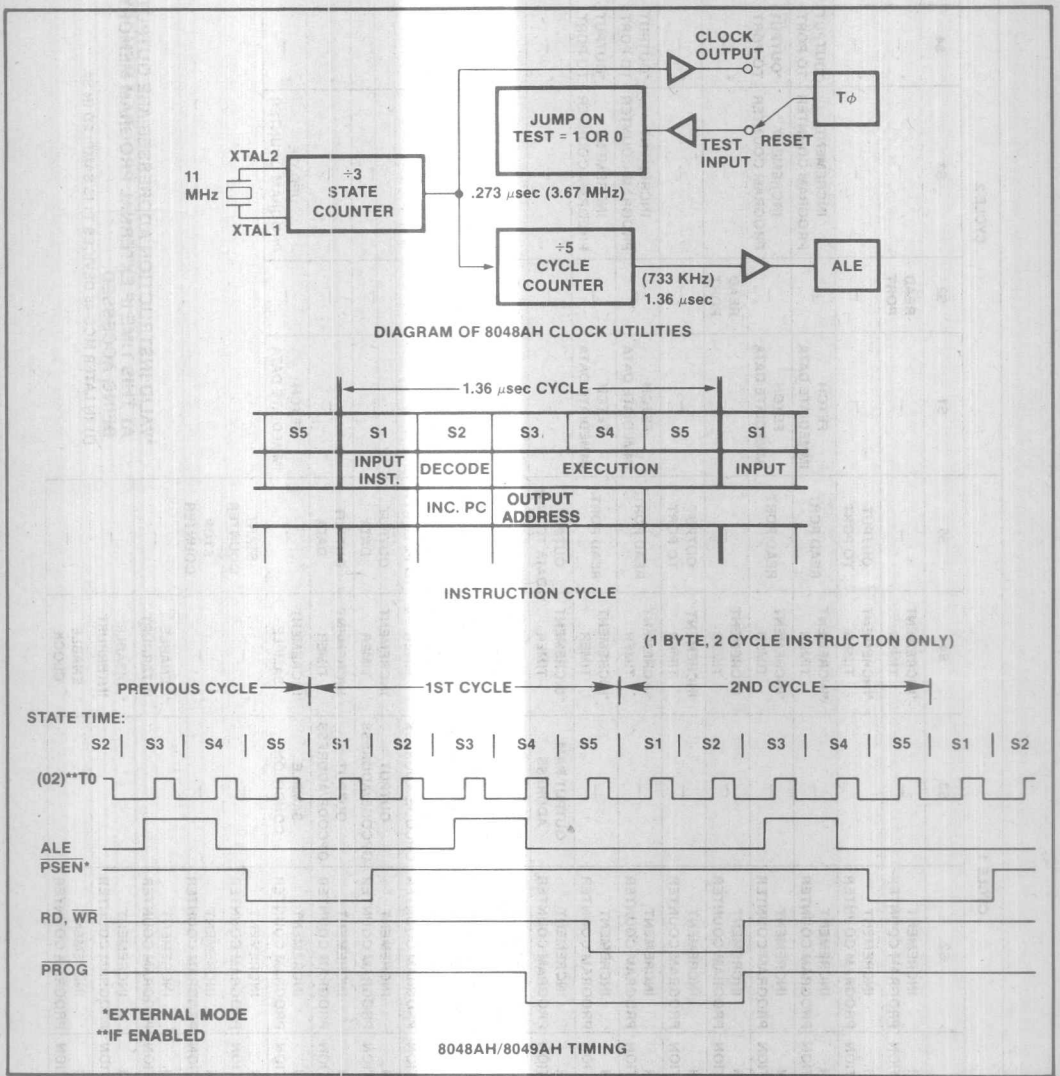


Figure 10. MCS®-48 Timing Generation and Cycle Timing

2.13 Single-Step

This feature, as pictured in Figure 13, provides the user with a debug capability in that the processor can be stepped through the program one instruction at a time. While stopped, the address of the next instruction to be fetched is available concurrently on BUS and the lower half of Port 2. The user can therefore follow the program through each of the instruction steps. A timing diagram, showing the interaction between output ALE and input $\overline{SS}$, is shown. The BUS buffer contents are lost during single step; however, a latch may be added to reestablish the lost I/O capability if needed. Data is valid at the leading edge of ALE.

	CYCLE 1					CYCLE 2				
INSTRUCTION	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
IN A,P	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	—	—	READ PORT	—	* —	—
OUTL P,A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	OUTPUT TO PORT	—	—	—	* —	—
ANL P, = DATA	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	INCREMENT PROGRAM COUNTER	*OUTPUT TO PORT	—
ORL P, = DATA	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	INCREMENT PROGRAM COUNTER	*OUTPUT TO PORT	—
INS A, BUS	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	—	—	READ PORT	—	* —	—
OUTL BUS, A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	OUTPUT TO PORT	—	—	—	* —	—
ANL BUS, = DATA	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	INCREMENT PROGRAM COUNTER	*OUTPUT TO PORT	—
ORL BUS, = DATA	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	INCREMENT PROGRAM COUNTER	*OUTPUT TO PORT	—
MOVX @ R,A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT RAM ADDRESS	INCREMENT TIMER	OUTPUT DATA TO RAM	—	—	—	* —	—
MOVX A,@R	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT RAM ADDRESS	INCREMENT TIMER	—	—	READ DATA	—	* —	—
MOVD A,P_i	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT OPCODE/ADDRESS	INCREMENT TIMER	—	—	READ P2 LOWER	—	* —	—
MOVD P_i,A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT OPCODE/ADDRESS	INCREMENT TIMER	OUTPUT DATA TO P2 LOWER	—	—	—	* —	—
ANLD P,A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT OPCODE/ADDRESS	INCREMENT TIMER	OUTPUT DATA	—	—	—	* —	—
ORLD P,A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT OPCODE/ADDRESS	INCREMENT TIMER	OUTPUT DATA	—	—	—	* —	—
J(CONDITIONAL)	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	SAMPLE CONDITION	*INCREMENT SAMPLE	—	FETCH IMMEDIATE DATA	—	UPDATE PROGRAM COUNTER	* —	—
STRT T STRT CNT	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	* —	START COUNTER					
STOP TCNT	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	* —	STOP COUNTER					
ENI	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	* ENABLE INTERRUPT	—					
DIS I	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	* DISABLE INTERRUPT	—					
ENTO CLK	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER		* ENABLE CLOCK	—					

*VALID INSTRUCTION ADDRESSES ARE OUTPUT AT THIS TIME IF EXTERNAL PROGRAM MEMORY IS BEING ACCESSED.

(1) IN LATER MCS-48 DEVICES T1 IS SAMPLED IN S4.

Figure 11. 8048AH/8049AH Instruction Timing Diagram

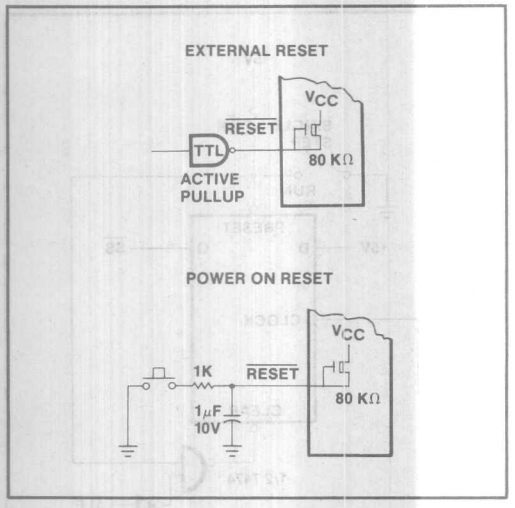


Figure 12.

TIMING

The 8048AH operates in a single-step mode as follows:

1) The processor is requested to stop by applying a low level on $\overline{SS}$.
2) The processor responds by stopping during the address fetch portion of the next instruction. If a double cycle instruction is in progress when the single step command is received, both cycles will be completed before stopping.
3) The processor acknowledges it has entered the stopped state by raising ALE high. In this state (which can be maintained indefinitely) the address of the next instruction to be fetched is present on BUS and the lower half of port 2.
4) $\overline{SS}$ is then raised high to bring the processor out of the stopped mode allowing it to fetch the next instruction. The exit from stop is indicated by the processor bringing ALE low.
5) To stop the processor at the next instruction $\overline{SS}$ must be brought low again soon after ALE goes low. If $\overline{SS}$ is left high the processor remains in a "Run" mode.

A diagram for implementing the single-step function of the 8748H is shown in Figure 13. D-type flip-flop with preset and clear is used to generate $\overline{SS}$. In the run mode SS is held high by keeping the flip-flop preset (preset has precedence over the clear input). To enter single step, preset is removed allowing ALE to bring $\overline{SS}$ low via the clear input. ALE should be buffered since the clear input of an SN7474 is the equivalent of 3 TTL loads. The processor is now in the stopped state. The next instruction is initiated by clocking a "1" into the flip-flop. This "1" will not appear on $\overline{SS}$ unless ALE is high removing clear from the flip-flop. In response to $\overline{SS}$ going high the processor begins an instruction fetch which brings ALE low resetting $\overline{SS}$ through the clear input and causing the processor to again enter the stopped state.

2.14 Power Down Mode (8048AH, 8049AH, 8050AH, 8039AHL, 8035AHL, 8040AHL)

Extra circuitry has been added to the 8048AH/8049AH/8050AH ROM version to allow power to be removed from all but the data RAM array for low power standby operation. In the power down mode the contents of data RAM can be maintained while drawing typically 10% to 15% of normal operating power requirements.

V_{CC} serves as the 5V supply pin for the bulk of circuitry while the V_{DD} pin supplies only the RAM array. In normal operation both pins are a 5V while in standby, V_{CC} is at ground and V_{DD} is maintained at its standby value. Applying Reset to the processor through the $\overline{RESET}$ pin inhibits any access to the RAM by the processor and guarantees that RAM cannot be inadvertently altered as power is removed from V_{CC}.

A typical power down sequence (Figure 14) occurs as follows:

1) Imminent power supply failure is detected by user defined circuitry. Signal must be early enough to allow 8048AH to save all necessary data before V_{CC} falls below normal operating limits.
2) Power fail signal is used to interrupt processor and vector it to a power fail service routine.
3) Power fail routine saves all important data and machine status in the internal data RAM array. Routine may also initiate transfer of backup supply to the V_{DD} pin and indicate to external circuitry that power fail routine is complete.
4) Reset is applied to guarantee data will not be altered as the power supply falls out of limits. Reset must be held low until V_{CC} is at ground level.

Recovery from the Power Down mode can occur as any other power-on sequence with an external capacitor on the Reset input providing the necessary delay. See the previous section on Reset.

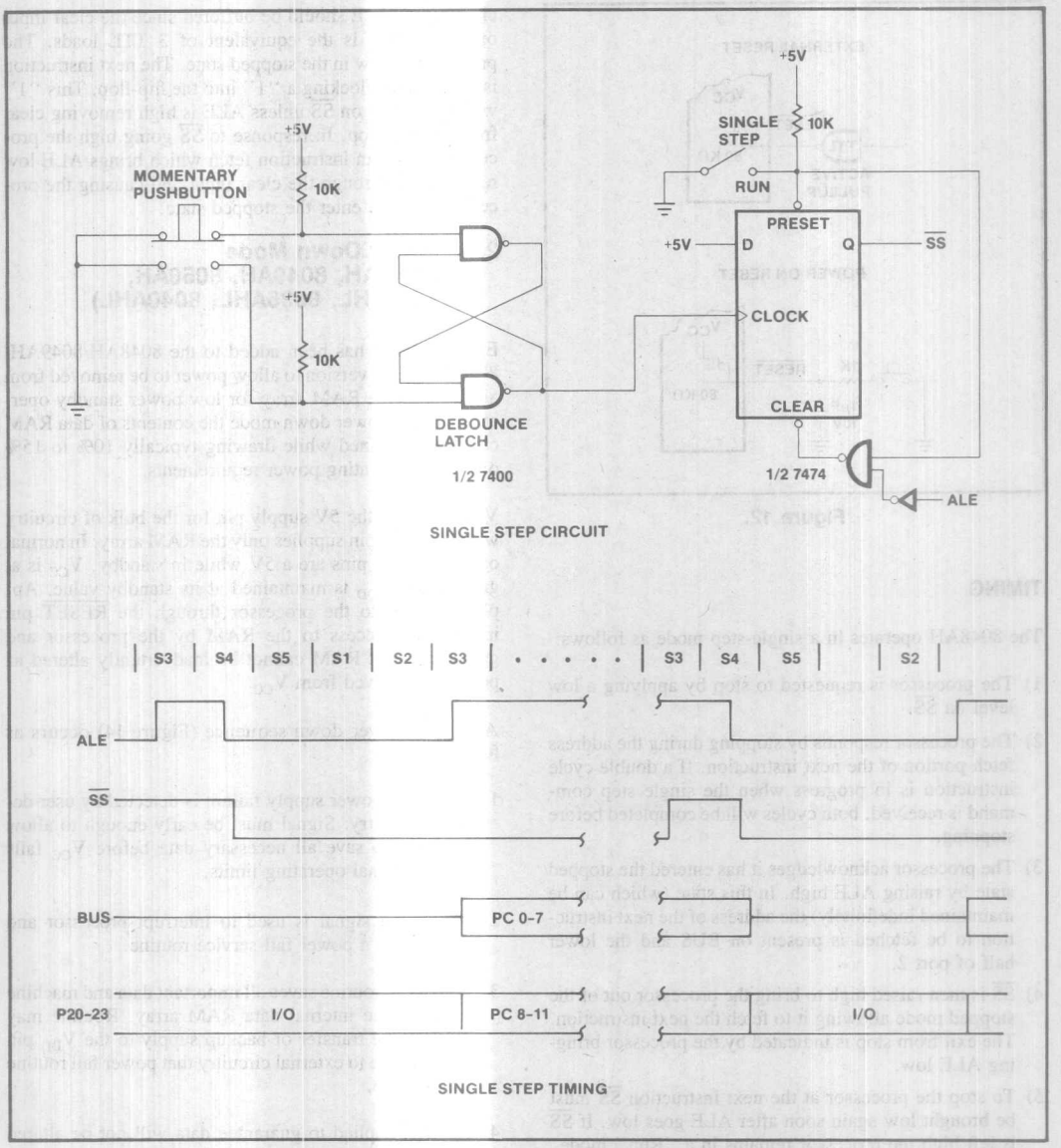


Figure 13. Single Step Operation

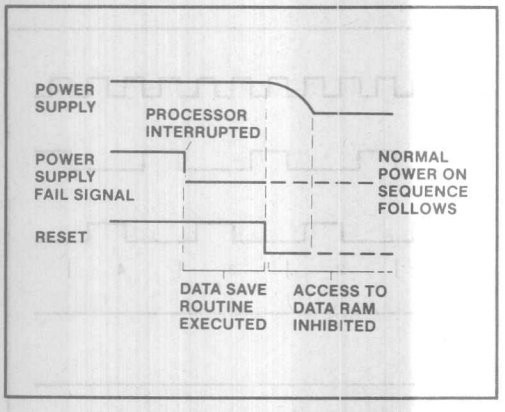


Figure 14. Power Down Sequence

2.15 External Access Mode

Normally the first 1K (8048AH), 2K (8049AH), or 4K (8050AH) words of program memory are automatically fetched from internal ROM or EPROM. The EA input pin however allows the user to effectively disable internal program memory by forcing all program memory fetches to reference external memory. The following chapter explains how access to external program memory is accomplished.

The External Access mode is very useful in system test and debug because it allows the user to disable his internal applications program and substitute an external program of his choice — a diagnostic routine for instance. In addition, the date sheet shows how internal program memory can be read externally, independent of the processor. A "1" level on EA initiates the external accesss mode. For proper operation, Reset should be applied while the EA input is changed.

2.16 Sync Mode

The 8048AH, 8049AH, 8050AH has incorporated a new SYNC mode. The Sync mode is provided to ease the design of multiple controller circuits by allowing the designer to force the device into known phase and state time. The SYNC mode may also be utilized by automatic test equipment (ATE) for quick, easy, and efficient synchronizing between the tester and the DUT (device under test).

SYNC mode is enabled when SS′ pin is raised to high voltage level of +12 volts. To begin synchronization, T0 is raised to 5 volts at least four clocks cycles after SS′. T0 must be high for at least four X1 clock cycles to fully reset the prescaler and time state generators. T0 may then be brought down with the rising edge of X1. Two clock cycles later, with the rising edge of X1, the device enters into Time State 1, Phase 1, SS′ is then brought down to 5 volts 4 clocks later after T0. RESET′ is allowed to go high 5 tCY (75 clocks) later for normal execution of code. See Figure 15.

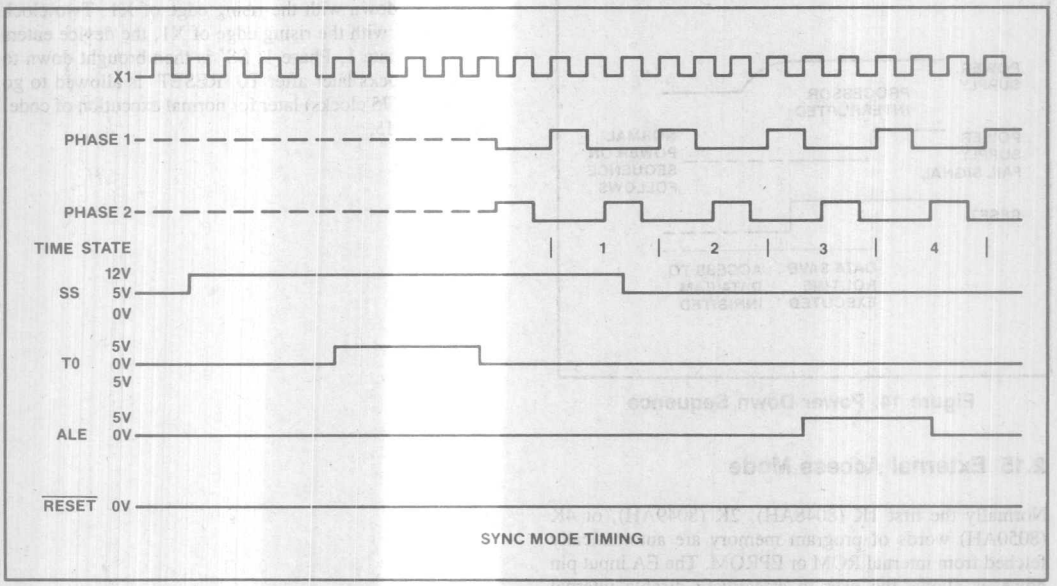


Figure 15. Sync Mode Timing

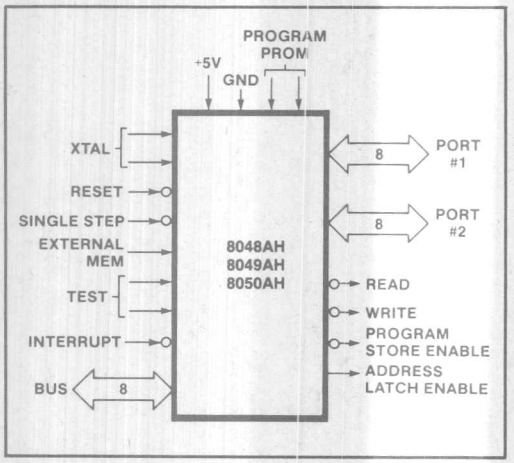


Figure 16. 8048AH and 8049AH Logic Symbol

3.0 PIN DESCRIPTION

The MCS-48 processors are packaged in 40 pin Dual In-Line Packages (DIP's). Table 3 is a summary of the functions of each pin. Figure 16 is the logic symbol for the 8048AH product family. Where it exists, the second paragraph describes each pin's function in an expanded MCS-48 system. Unless otherwise specified, each input is TTL compatible and each output will drive one standard TTL load.

Table 3. Pin Description

Designation	Pin Number*	Function
V_{SS}	20	Circuit GND potential
V_{DD}	26	Programming power supply; 21V during program for the 8748H/8749H; +5V during operation for both ROM and EPROM. Low power standby pin in 8048AH and 8049AH/8050AH ROM versions.
V_{CC}	40	Main power supply; +5V during operation and during 8748H and 8749H programming.
PROG	25	Program pulse; +18V input pin during 8748H/8749H programming. Output strobe for 8243 I/O expander.
P10–P17 (Port 1)	27-34	8-bit quasi-bidirectional port. (Internal Pullup $\approx$ 50KΩ)
P20–P27 (Port 2)	21–24 35–38	8-bit quasi-bidirectional port. (Internal Pullup $\approx$ 50KΩ) P20–P23 contain the four high order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for 8243.
D0–D7 (BUS)	12–19	True bidirectional port which can be written or read synchronously using the $\overline{RD}$, $\overline{WR}$ strobes. The port can also be statically latched. Contains the 8 low order program counter bits during an external program memory fetch, and receives the addressed instruction under the control of $\overline{PSEN}$. Also contains the address and data during an external RAM data store instruction, under control of ALE, $\overline{RD}$, and $\overline{WR}$.
T0	1	Input pin testable using the conditional transfer instructions JT0 and JNT0. T0 can be designated as a clock output using ENTO CLK instruction. T0 is also used during programming and sync mode.
T1	39	Input pin testable using the JT1, and JNT1 instructions. Can be designated the event counter input using the STRT CNT instruction. (See Section 2.10).
$\overline{INT}$	6	Interrupt input. Initiates an interrupt if interrupt is enabled. Interrupt is disabled after a reset. (Active low) Interrupt must remain low for at least 3 machine cycles to ensure proper operation.
$\overline{RD}$	8	Output strobe activated during a BUS read. Can be used to enable data onto the BUS from an external device. (Active low) Used as a Read Strobe to External Data Memory.
$\overline{RESET}$	4	Input which is used to initialize the processor. Also used during EPROM programming and verification. (Active low) (Internal pullup $\approx$ 80K Ω)
$\overline{WR}$	10	Output strobe during a BUS write. (Active low) Used as write strobe to external data memory.
ALE	11	Address Latch Enable. This signal occurs once during each cycle and is useful as a clock output. The negative edge of ALE strobes address into external data and program memory.

Table 3. Pin Description (Continued)

Designation	Pin Number*	Function
$\overline{PSEN}$	9	Program Store Enable. This output occurs only during a fetch to external program memory. (Active low)
$\overline{SS}$	5	Single step input can be used in conjunction with ALE to "single step" the processor through each instruction. (Active low) (Internal pullup ≈ 300KΩ) +12V for sync modes (See 2.16).
EA	7	External Access input which forces all program memory fetches to reference external memory. Useful for emulation and debug, and essential for testing and program verification. (Active high) +12V for 8048AH/8049AH/8050AH program verification and +18V for 8748H/8749H program verification (Internal pullup ≈ 10MΩ on 8048AH/8049AH/8035AHL/8039AHL/8050AH/8040AHL)
XTAL1	2	One side of crystal input for internal oscillator. Also input for external source.
XTAL2	3	Other side of crystal/external source input.

*Unless otherwise stated, inputs do not have internal pullup resistors. 8048AH, 8748H, 8049AH, 8050AH, 8040AHL

4.0 PROGRAMMING, VERIFYING AND ERASING EPROM

The internal Program Memory of the 8748H and the 8749H may be erased and reprogrammed by the user as explained in the following sections. See also the 8748H and 8749H data sheets.

4.1 Programming/Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. This programming algorithm applies to both the 8748H and 8749H. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a descsription of their functions:

Pin	Function
XTAL 1	Clock Input (3 to 4 MHz)
Reset	Initialization and Address Latching
Test 0	Selection of Program (0V) or Verify (5V) Mode
EA	Activation of Program/Verify Modes
BUS	Address and Data Input Data Output During Verify
P20-1	Address Input for 8748H
P20-2	Address Input for 8749H
V_{DD}	Programming Power Supply
PROG	Program Pulse Input
P10–P11	Tied to ground (8749H only)

8748H AND 8749H ERASURE CHARACTERISTICS

The erasure characteristics of the 8748H and 8749H are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (A). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000A range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8748H and 8749H in approximately 3 years while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 8748H or 8749H is to be exposed to these types of lighting conditions for extended periods of time, opaque labels should be placed over the 8748H window to prevent unintentional erasure.

When erased, bits of the 8748H and 8749H Program Memory are in the logic "0" state.

The recommended erasure procedure for the 8748H and 8749H is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (A). The integrated dose (i.e., UV intensity X exposure time) for erasure should be a minimum of 15W-sec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000μW/cm^2 power rating. The 8748H and 8749H should be placed within one inch from the lamp tubes during erasure. Some lamps have a filter in their tubes and this filter should be removed before erasure.

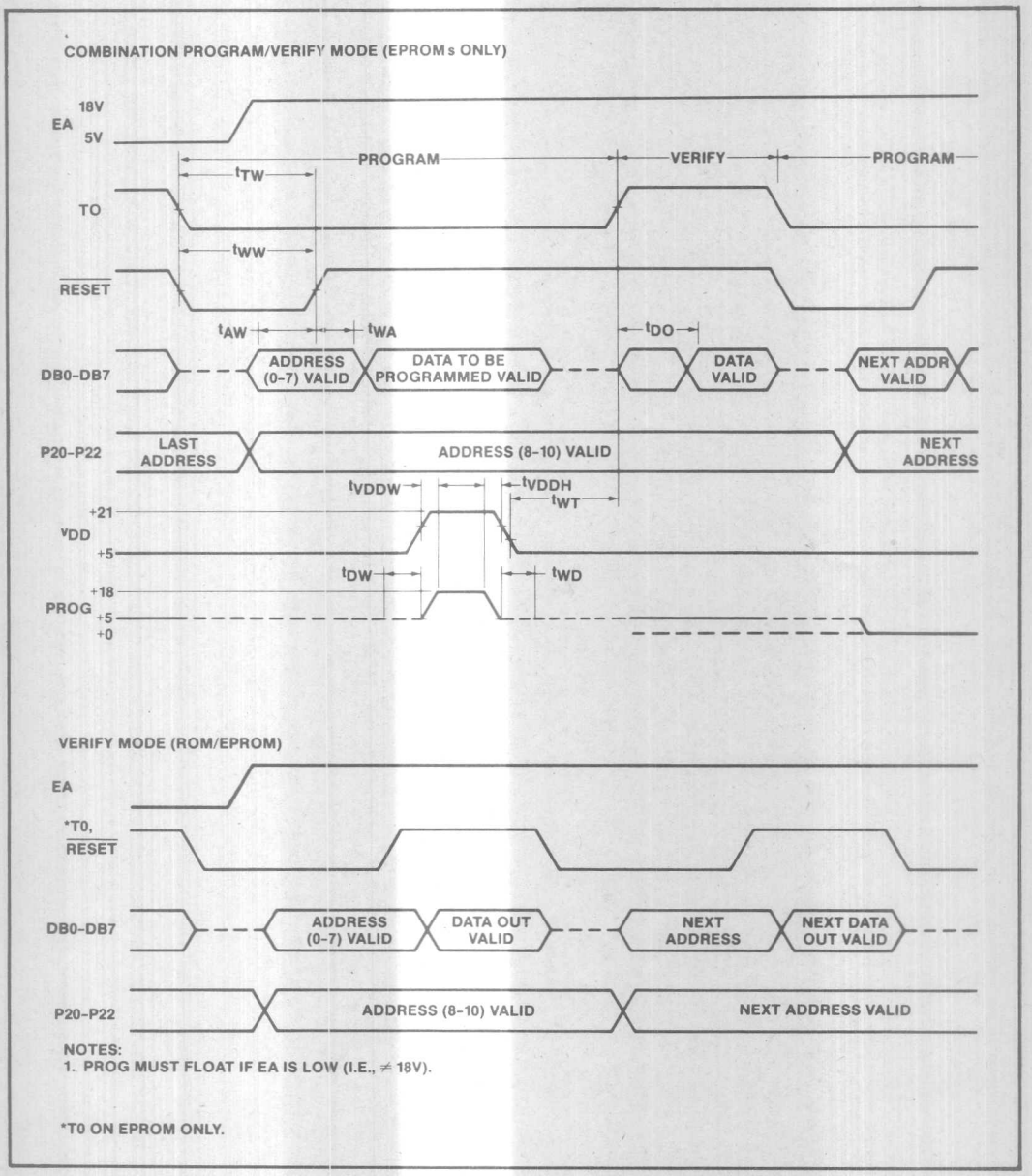
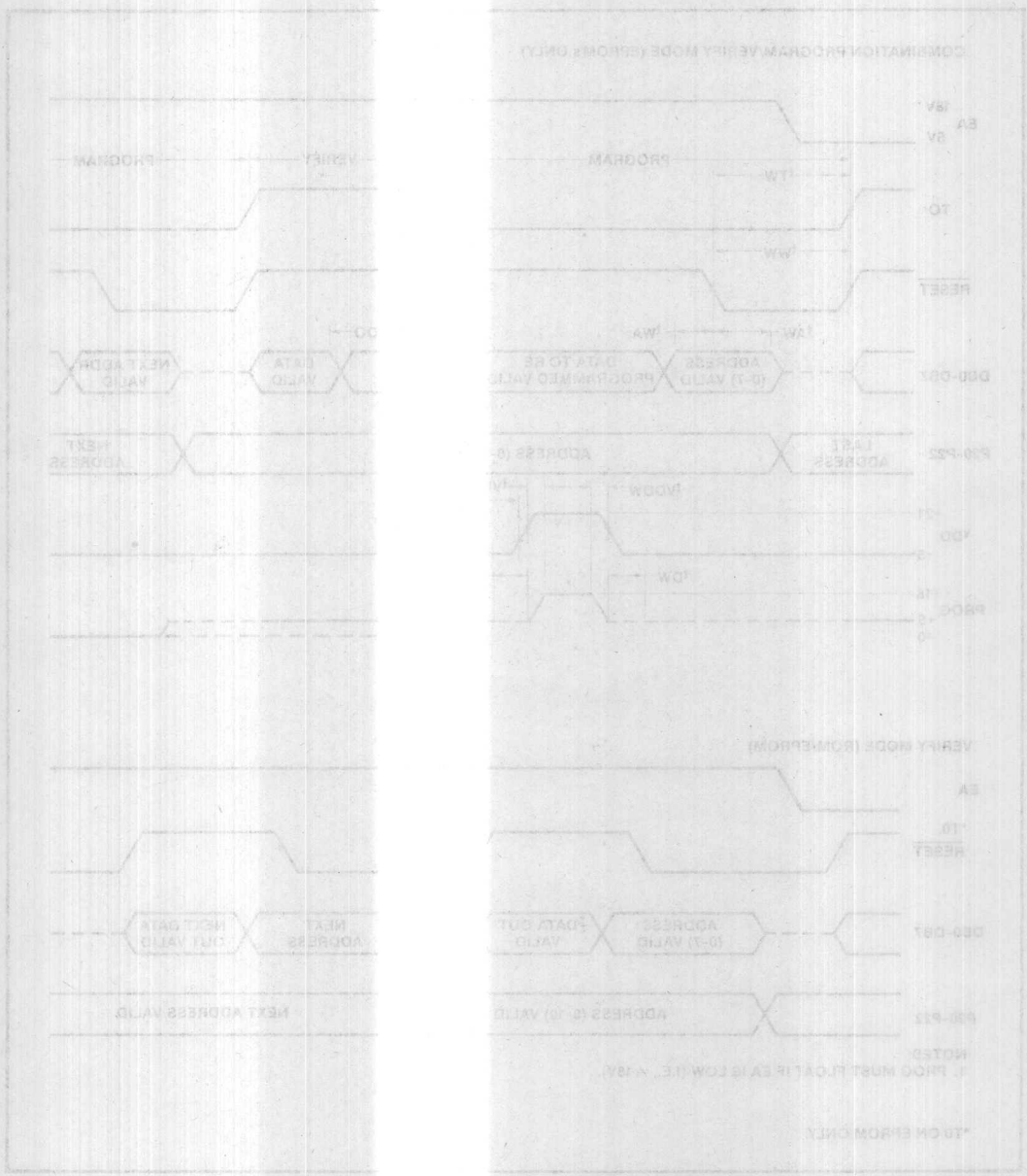


Figure 17. Program/Verify Sequence for 8749H/8748H



MCS®-48 Expanded System 2

EXPANDED MCS®-48 SYSTEM

1.0 INTRODUCTION

If the capabilities resident on the single-chip 8048AH/8748H/8035AHL/8049AH/8749H/8039AHL are not sufficient for your system requirements, special on-board circuitry allows the addition of a wide variety of external memory, I/O, or special peripherals you may require. The processors can be directly and simply expanded in the following areas:

- Program Memory to 4K words
- Data Memory to 320 words (384 words with 8049AH)
- I/O by unlimited amount
- Special Functions using 8080/8085AH peripherals

By using bank switching techniques, maximum capability is essentially unlimited. Bank switching is discussed later in the chapter. Expansion is accomplished in two ways:

1) Expander I/O — A special I/O Expander circuit, the 8243, provides for the addition of four 4-bit Input/Output ports with the sacrifice of only the lower half (4-bits) of port 2 for inter-device communication. Multiple 8243's may be added to this 4-bit bus by generating the required "chip select" lines.
2) Standard 8085 Bus — One port of the 8048AH/8049AH is like the 8-bit bidirectional data bus of the 8085 microcomputer system allowing interface to the numerous standard memories and peripherals of the MCS®-80/85 microcomputer family.

MCS-48 systems can be configured using either or both of these expansion features to optimize system capabilities to the application.

Both expander devices and standard memories and peripherals can be added in virtually any number and combination required.

2.0 EXPANSION OF PROGRAM MEMORY

Program Memory is expanded beyond the resident 1K or 2K words by using the 8085 BUS feature of the MCS®-48. All program memory fetches from the addresses less than 1024 on the 8048AH and less than 2048 on the 8049AH occur internally with no external signals being generated (except ALE which is always present). At address 1024 on the 8048AH, the processor automatically initiates external program memory fetches.

2.1 Instruction Fetch Cycle (External)

As shown in Figure 1, for all instruction fetches from addresses of 1024 (2048) or greater, the following will occur:

1) The contents of the 12-bit program counter will be output on BUS and the lower half of port 2.
2) Address Latch Enable (ALE) will indicate the time at which address is valid. The trailing edge of ALE is used to latch the address externally.
3) Program Store Enable ($\overline{PSEN}$) indicates that an external instruction fetch is in progress and serves to enable the external memory device.
4) BUS reverts to input (floating) mode and the processor accepts its 8-bit contents as an instruction word.

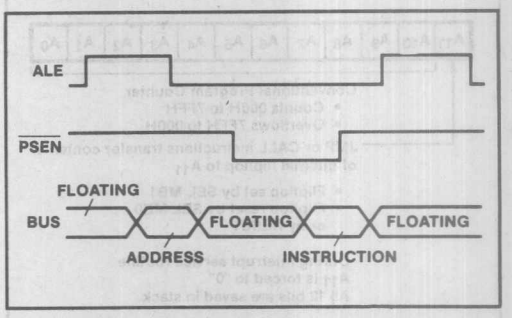


Figure 1. Instruction Fetch from External Program Memory

All instruction fetches, including internal addresses, can be forced to be external by activating the EA pin of the 8048AH/8049AH/8050AH. The 8035AHL/8039AHL/8040AHL processors without program memory always operate in the external program memory mode (EA = 5V).

2.2 Extended Program Memory Addressing (Beyond 2K)

For programs of 2K words or less, the 8048AH/8049AH addresses program memory in the conventional manner. Addresses beyond 2047 can be reached by executing a program memory bank switch instruction (SEL MB0, SEL MB1) followed by a branch instruction (JMP or CALL). The bank switch feature extends the range of branch instructions beyond their normal 2K range and at the same time prevents the user from inadvertently crossing the 2K boundary.

PROGRAM MEMORY BANK SWITCH

The switching of 2K program memory banks is accomplished by directly setting or resetting the most significant bit of the program counter (bit 11); see Figure 2. Bit 11 is not altered by normal incrementing of the program counter but is loaded with the contents of a special flip-flop each time a JMP or CALL instruction is executed. This special flip-flop is set by executing an SEL MB1

instruction and reset by SEL MB0. Therefore, the SEL MB instruction may be executed at any time prior to the actual bank switch which occurs during the next branch instruction encountered. Since all twelve bits of the program counter, including bit 11, are stored in the stack, when a Call is executed, the user may jump to subroutines across the 2K boundary and the proper bank will be restored upon return. However, the bank switch flip-flop will not be altered on return.

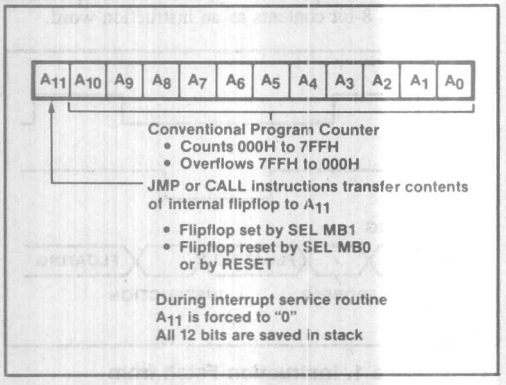


Figure 2. Program Counter

INTERRUPT ROUTINES

Interrupts always vector the program counter to location 3 or 7 in the first 2K bank, and bit 11 of the program counter is held at "0" during the interrupt service routine. The end of the service routine is signalled by the execution of an RETR instruction. Interrupt service routines should therefore be contained entirely in the lower 2K words of program memory. The execution of a SEL MB0 or SEL MB1 instruction within an interrupt routine is not recommended since it will not alter PC11 while in the routine, but will change the internal flip-flop.

2.3 Restoring I/O Port Information

Although the lower half of Port 2 is used to output the four most significant bits of address during an external program memory fetch, the I/O information is still outputed during certain portions of each machine cycle. I/O information is always present on Port 2's lower 4 bits at the rising edge of ALE and can be sampled or latched at this time.

2.4 Expansion Examples

Shown in Figure 3 is the addition of 2K words of program memory using an 2716A 2K x 8 ROM to give a total of 3K words of program memory. In this case no chip select decoding is required and $\overline{PSEN}$ enables the memory directly through the chip select input. If the system requires only 2K of program memory, the same configuration can be used with an 8035AHL substituted for the 8048AH. The 8049AH would provide 4K of program memory with the same configuration.

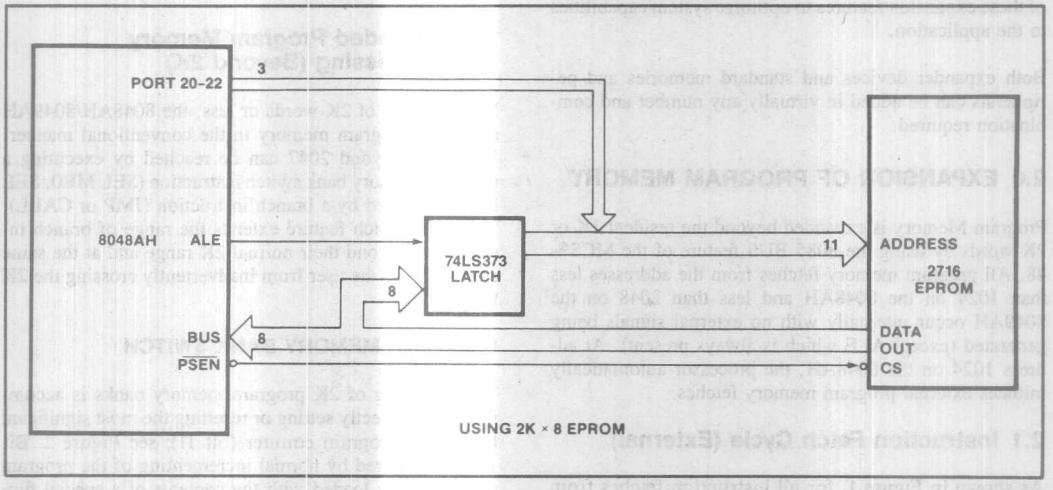


Figure 3. Expanding MCS®-48 Program Memory Using Standard Memory Products

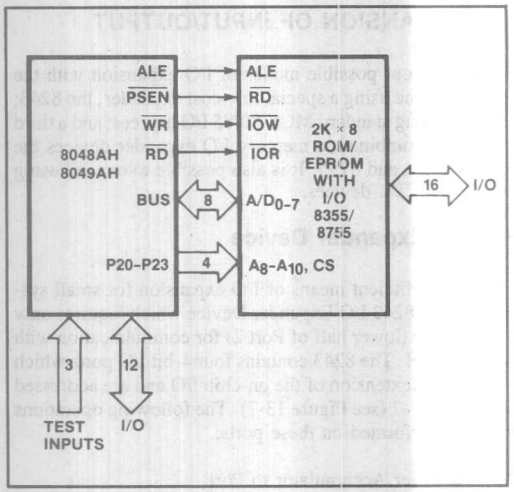


Figure 4. External Program Memory Interface

Figure 4 shows how the 8755/8355 EPROM/ROM with I/O interfaces directly to the 8048AH without the need for an address latch. The 8755/8355 contains an internal 8-bit address latch eliminating the need for an 8212 latch. In addition to a 2K x 8 program memory, the 8755/8355 also contains 16 I/O lines addressable as two 8-bit ports. These ports are addressed as external RAM; therefore the $\overline{RD}$ and $\overline{WR}$ outputs of the 8048AH are required. See the following section on data memory expansion for more detail. The subsequent section on I/O expansion explains the operation of the 16 I/O lines.

3.0 EXPANSION OF DATA MEMORY

Data Memory is expanded beyond the resident 64 words by using the 8085AH type bus feature of the MCS®-48.

3.1 Read/Write Cycle

All address and data is transferred over the 8 lines of BUS. As shown in Figure 5, a read or write cycle occurs as follows:

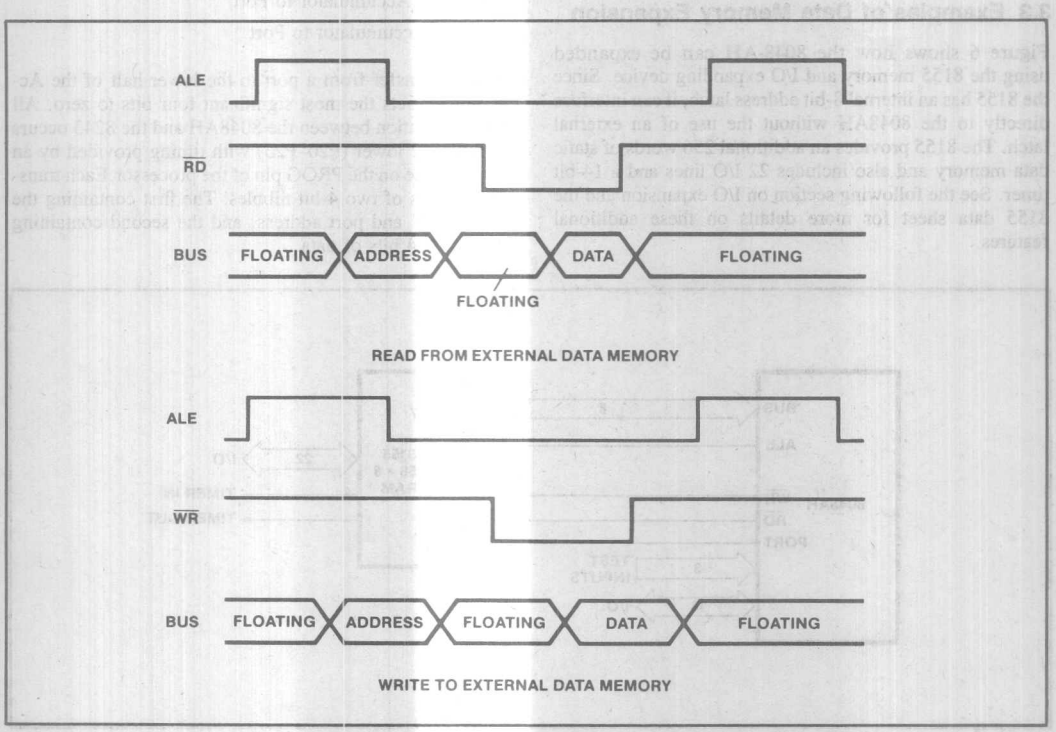


Figure 5. External Data Memory Timings

1) The contents of register R0 or R1 is outputed on BUS.
2) Address Latch Enable (ALE) indicates addresss is valid. The trailing edge of ALE is used to latch the address externally.
3) A read ($\overline{RD}$) or write ($\overline{WR}$) pulse on the corresponding output pins of the 8048AH indicates the type of data memory access in progress. Output data is valid at the trailing edge of $\overline{WR}$ and input data must be valid at the trailing edge of $\overline{RD}$.
4) Dat (8 bits) is transferred in or out over BUS.

3.2 Addressing External Data Memory

External Data Memory is accessed with its own two-cycle move instructions. MOVXA, @R and MOVX@R, A, which transfer 8 bits of data between the accumulator and the external memory location addressed by the contents of one of the RAM Pointer Registers R0 and R1. This allows 256 locations to be addressed in addition to the resident locations. Additional pages may be added by "bank switching" with extra output lines of the 8048AH.

3.3 Examples of Data Memory Expansion

Figure 6 shows how the 8048-AH can be expanded using the 8155 memory and I/O expanding device. Since the 8155 has an internal 8-bit address latch, it can interface directly to the 8048AH without the use of an external latch. The 8155 provides an additional 256 words of static data memory and also includes 22 I/O lines and a 14-bit timer. See the following section on I/O expansion and the 8155 data sheet for more details on these additional features.

4.0 EXPANSION OF INPUT/OUTPUT

There are four possible modes of I/O expansion with the 8048AH: one using a special low-cost expander, the 8243; another using standard MCS-80/85 I/O devices; and a third using the combination memory I/O expander devices the 8155, 8355, and 8755. It is also possible to expand using standard TTL devices.

4.1 I/O Expander Device

The most efficient means of I/O expansion for small systems is the 8243 I/O Expander Device which requires only 4 port lines (lower half of Port 2) for communication with the 8048AH. The 8243 contains four 4-bit I/O ports which serve as an extension of the on-chip I/O and are addressed as ports #4–7 (see Figure 13-7). The following operations may be performed on these ports:

- Transfer Accumulator to Port
- Transfer Port to Accumulator
- AND Accumulator to Port
- OR Accumulator to Port

A 4-bit transfer from a port to the lower half of the Accumulator sets the most significant four bits to zero. All communication between the 8048AH and the 8243 occurs over Port 2 lower (P20–P23) with timing provided by an output pulse on the PROG pin of the processor. Each transfer consists of two 4-bit nibbles: The first containing the "op code" and port address, and the second containing the actual 4 bits of data.

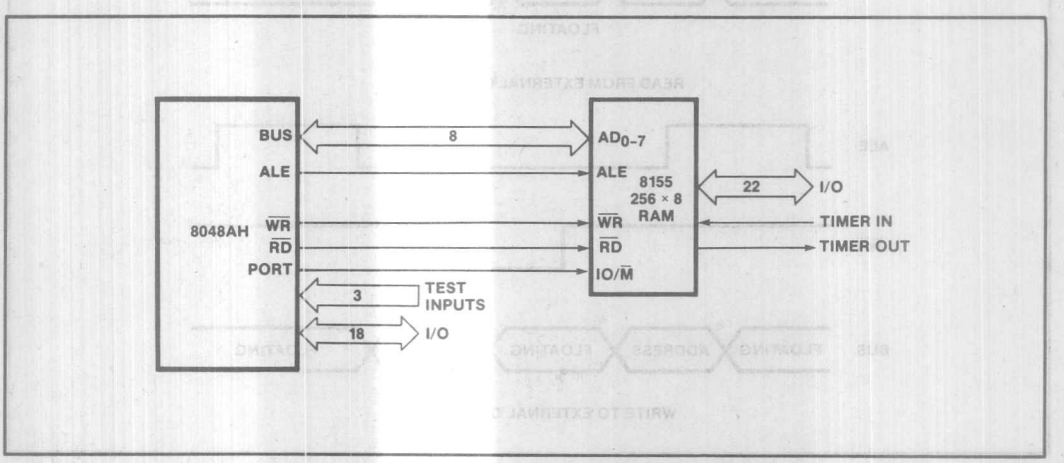


Figure 6. 8048AH Interface to 256 x 8 Standard Memories

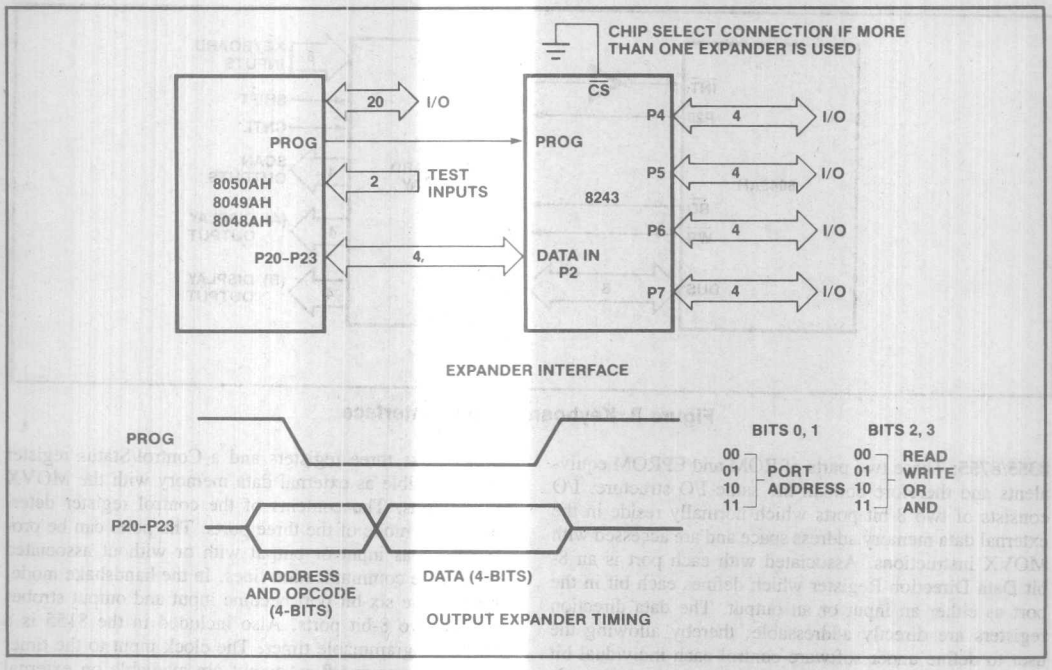


Figure 7. 8243 Expander I/O Interface



II
00 Read
01 Write
10 OR
11 AND

AA
00 — Port #4
01 — Port #5
10 — Port #6
11 — Port #7

A high to low transition of the PROG line indicates that address is present, while allow to high transition indicates the presence of data. Additional 8243's may be added to the four-bit bus and chip selected using additional output lines from the 8048AH/8748H.

I/O PORT CHARACTERISTICS

Each of the four 4-bit ports of the 8243 can serve as either input or output and can provide high drive capability in both the high and low state.

4.2 I/O Expansion with Standard Peripherals

Standard MCS-80/85 type I/O devices may be added to the MCS®-48 using the same bus and timing used for Data Memory expansion. Figure 8 shows an example of how an 8048AH can be connected to an MCS-85 peripheral. I/O devices reside on the Data Memory bus and in the data memory address space and are accessed with the same MOVX instructions. (See the previous section on data memory expansion for a description of timing.) The following are a few of the Standard MCS-80 devices which are very useful in MCS®-48 systems:

- 8214 Priority Interrupt Encoder
- 8251 Serial Communications Interface
- 8255 General Purpose Programmable I/O
- 8279 Keyboard/Display Interface
- 8254 Interval Timer

4.3 Combination Memory and I/O Expanders

As mentioned in the sections on program and data memory expansion, the 8355/8755 and 8155 expanders also contain I/O capability.

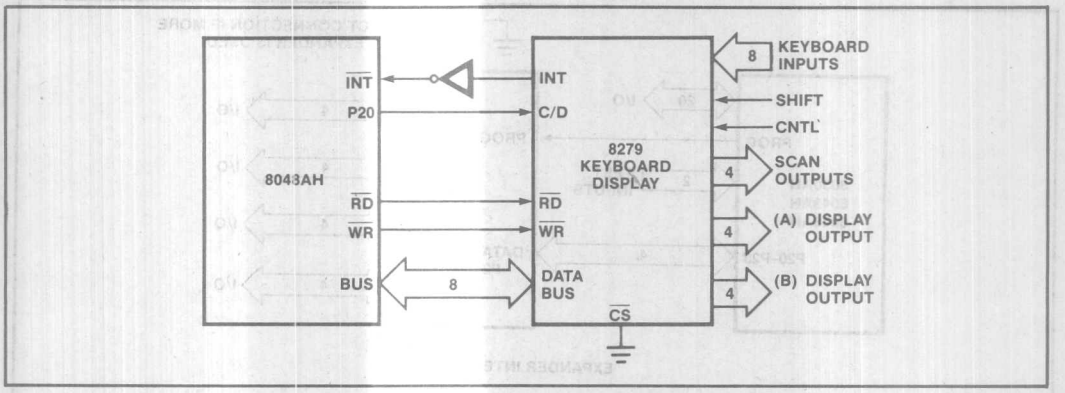


Figure 8. Keyboard/Display Interface

8355/8755: These two parts of ROM and EPROM equivalents and therefore contain the same I/O structure. I/O consists of two 8-bit ports which normally reside in the external data memory address space and are accessed with MOVX instructions. Associated with each port is an 8-bit Data Direction Register which defines each bit in the port as either an input or an output. The data direction registers are directly addressable, thereby allowing the user to define under software control each individual bit of the ports as either input or output. All outputs are statically latched and double buffered. Inputs are not latched.

8155/8156: I/O on the 8155/8156 is configured as two 8-bit programmable I/O ports and one 6-bit programmable port. These three registers and a Control/Status register are accessible as external data memory with the MOVX instructions. The contents of the control register determines the mode of the three ports. The ports can be programmed as input or output with or without associated handshake communication lines. In the handshake mode, lines of the six-bit port become input and output strobes for the two 8-bit ports. Also included in the 8155 is a 14-bit programmable timer. The clock input to the timer and the timer overflow output are available on external pins. The timer can be programmed to stop on terminal count or to continuously reload itself. A square wave or pulse output on terminal count can also be specified.

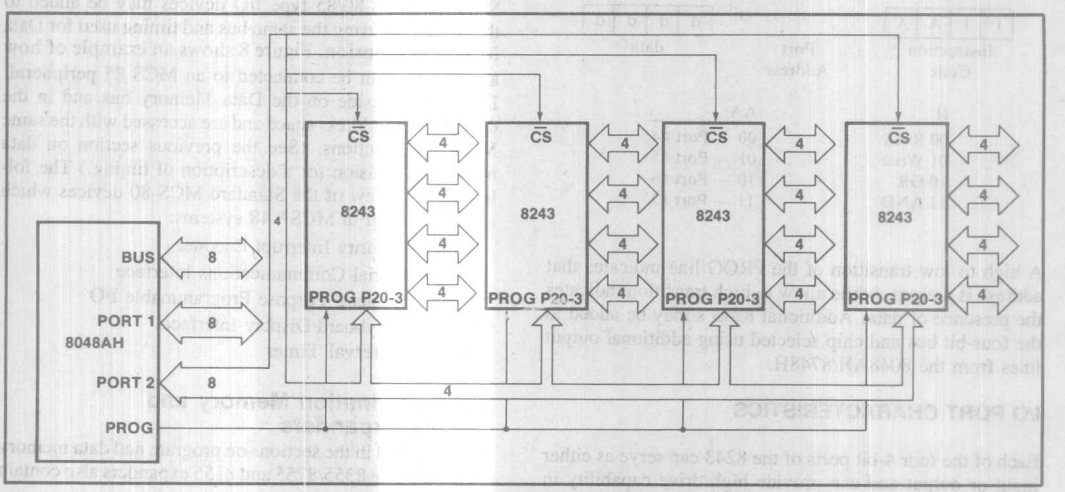


Figure 9. Low Cost I/O Expansion

I/O EXPANSION EXAMPLES

Figure 9 shows the expansion of I/O using multiple 8243's. The only difference from a single 8243 system is the addition of chip selects provided by additional 8048AH output lines. Two output liens and a decoder could also be used to address the four chips. Large numbers of 8243's would require a chip select decoder chip such as the 8205 to save I/O pins.

Figure 10 shows the 8048AH interface to a standard MCS®-80 peripheral; in this case, the 8255 Programmable Peripheral Interface, a 40-pin part which provides three 8-bit programmable I/O ports. The 8255 bus interface is typical of programmable MCS®-80 peripherals with an 8-bit bidirectional data bus, a RD and WR input for Read/Write control, a CS (chip select) input used to enable the Read/Write control logic and the address inputs used to select various internal registers.

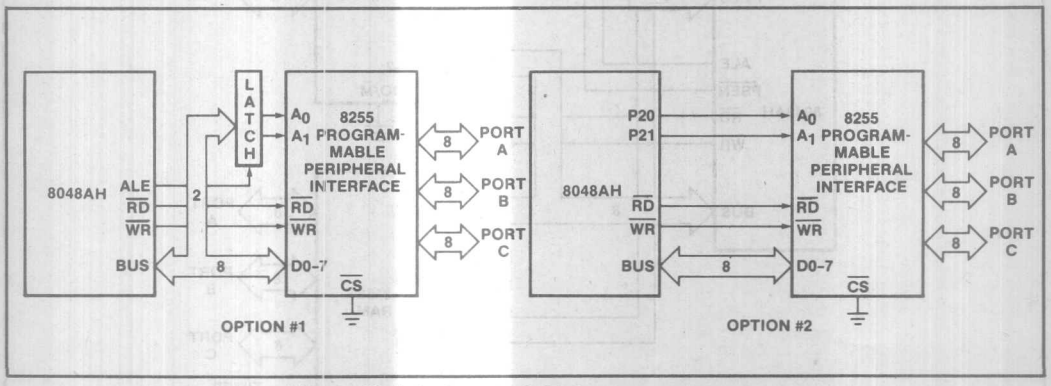


Figure 10. Interface to MCS®-80 Peripherals

Interconnection to the 8048AH is very straightforward with BUS, $\overline{RD}$, and $\overline{WR}$ connecting directly to the corresponding pins on the 8255. The only design consideration is the way in which the internal registers of the 8255 are to be addressed. If the registers are to be addressed as external data memory using the MOVX instructions, the appropriate number of address bits (in this case, 2) must be latched on BUS using ALE as described in the section on external data memories. If only a single device is connected to BUS, the 8255 may be continuously selected by grounding $\overline{CS}$. If multiple 8255's are used, additional address bits can be latched and used as chip selects.

A second addressing method eliminates external latches and chip select decoders by using output port lines as address and chip select lines directly. This method, of course, requires the setting of an output port with address information prior to executing a MOVX instruction.

5.0 MULTI-CHIP MCS®-48 SYSTEMS

Figure 11 shows the addition of two memory expanders to the 8048AH, one 8355/8755 ROM and one 8156 RAM. The main consideration in designing such a system is the addressing of the various memories and I/O ports. Note that in this configuration address lines A_{10} and A_{11} have been ORed to chip select the 8355. This ensures that the chip is active for all external program memory fetches in the 1K to 3K range and is disabled for all other addresses. This gating has been added to allow the I/O port of the 8355 to be used. If the chip was left selected all the time, there would be conflict between these ports and the RAM and I/O of the 8156. The NOR gate could be eliminated and A_{11} connected directly to the CE (instead of $\overline{CE}$) input of the 8355; however, this would create a 1K word "hole" in the program memory by causing the 8355 to be active in the 2K and 4K range instead of the normal 1K to 3K range.

In this system the various locations are addressed as follows:

- Data RAM — Addresses 0 to 255 when Port 2 Bit 0 has been previously set = 1 and Bit 1 set = 0
- RAM I/O — Addresses 0 to 3 when Port 2 Bit 0 = 1 and Bit 1 = 1
- ROM I/O — Addresses 0 to 3 when Port 2 Bit 2 or Bit 3 = 1

See the memory map in Figure 12.

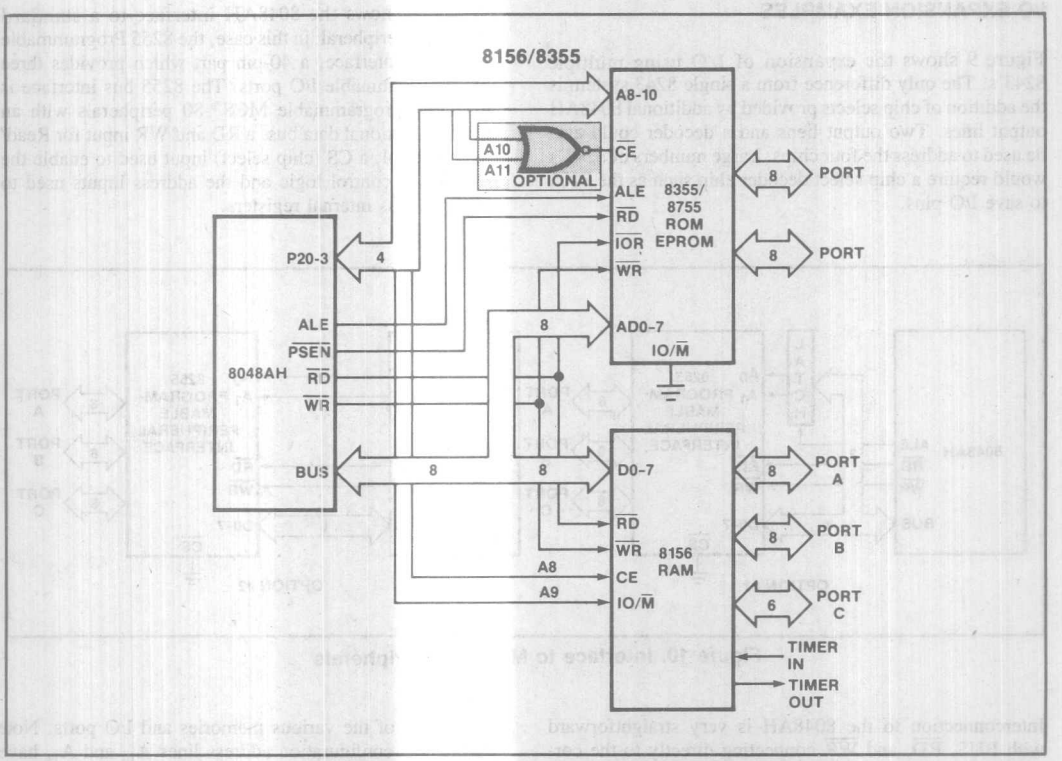


Figure 11. The Three-Component MCS®-48 System

6.0 MEMORY BANK SWITCHING

Certain systems may require more than the 4K words of program memory which are directly addressable by the program counter or more than the 256 data memory and I/O locations directly addressable by the pointer registers R0 and R1. These systems can be achieved using "bank switching" techniques. Bank switching is merely the selection of various blocks of "banks" of memory using dedicated output port lines from the processor. In the case of the 8048AH, program memory is selected in blocks of 4K words at a time, while data memory and I/O are enabled 256 words at a time.

The most important consideration in implementing two or more banks is the software required to cross the bank boundaries. Each crossing of the boundary requires that the processor first write a control bit to an output port before accessing memory or I/O in the new bank. If program memory is being switched, programs should be organized to keep boundary crossings to a minimum.

Jumping to subroutines across the boundary should be avoided when possible since the programmer must keep track of which bank to return to after completion of the subroutine. If these subroutines are to be nested and accessed from either bank, a software "stack" should be implemented to save the bank switch bit just as if it were another bit of the program counter.

From a hardware standpoint bank switching is very straightforward and involves only the connection of an I/O line or lines as bank enable signals. These enables are ANDed with normal memory and I/O chip select signals to activate the proper bank.

7.0 CONTROL SIGNAL SUMMARY

Table 1 summarizes the instructions which activate the various control outputs of the MCS®-48 processors. During all other instructions these outputs are driven to the active state.

Table 1. MCS®-48 Control Signals

Control Signal	When Active
$\overline{RD}$	During MOVX, A, @R or INS Bus
$\overline{WR}$	During MOVX @R, A or OUTL Bus
ALE	Every Machine Cycle
$\overline{PSEN}$	During Fetch of external program memory (instruction or immediate data)
PROG	During MOVD, A,P ANLD P,A MOVD P,A ORLD P,A

8.0 PORT CHARACTERISTICS

8.1 BUS Port Operations

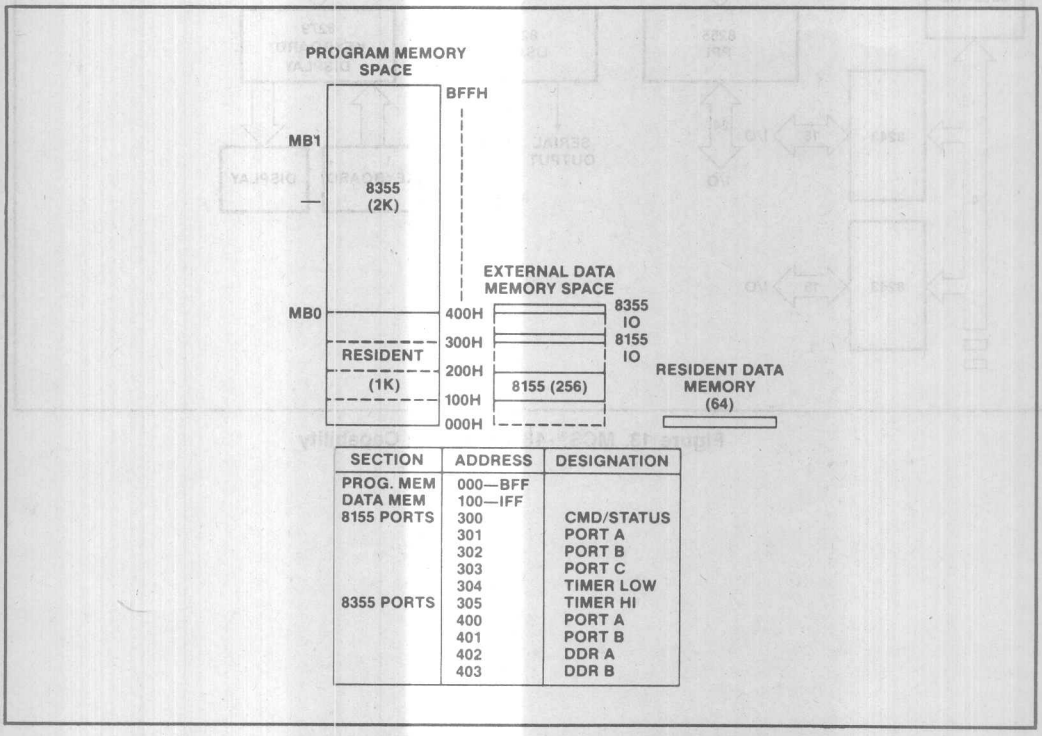
The BUS port can operate in three different modes: as a latched I/O port, as a bidirectional bus port, or as a program memory address output when external memory is used. The BUS port lines are either active high, active low, or high impedance (floating).

The latched mode (INS, OUTL) is intended for use in the single-chip configuration where BUS is not begin used as an expander port. OUTL and MOVX instructions can be mixed if necessary. However, a previously latched output will be destroyed by executing a MOVX instruction and BUS will be left in the high impedance state. INS does not put the BUS in a high impedance state. Therefore, the use of MOVX after OUTL to put the BUS in a high impedance state is necessary before an INS instruction intended to read an external word (as opposed to the previously latched value).

OUTL should never be used in a system with external program memory, since latching BUS can cause the next instruction, if external, to be fetched improperly.

8.2 Port 2 Operations

The lower half of Port 2 can be used in three different ways: as a quasi-bidirectional static port, as an 8243 expander port, and to adddress external program memory.



SECTION	ADDRESS	DESIGNATION
PROG. MEM	000—BFF	
DATA MEM	100—IFF	
8155 PORTS	300	CMD/STATUS
	301	PORT A
	302	PORT B
	303	PORT C
	304	TIMER LOW
8355 PORTS	305	TIMER HI
	400	PORT A
	401	PORT B
	402	DDR A
	403	DDR B

Figure 12. Memory Map for Three-Component MCS®-48 Family

In all cases outputs are driven low by an active device and driven high momentarily by a low impedance device and held high by a high impedance device to VCC.

The port may contain latched I/O data prior to its use in another mode without affecting operation of either. If lower Port 2 (P20-3) is used to output address for an external program memory fetch, the I/O information previously latched will be automatically removed temporarily while address is present, then retored when the fetch is complete. However, if lower Port 2 is used to communicate with an 8243, previously latched I/O information will be removed and not restored. After an input from the 8243, P20-3 will be left in the input mode (floating). After an output to the 8243, P20-3 will contain the value written, ANDed, or ORed to the 8243 port.

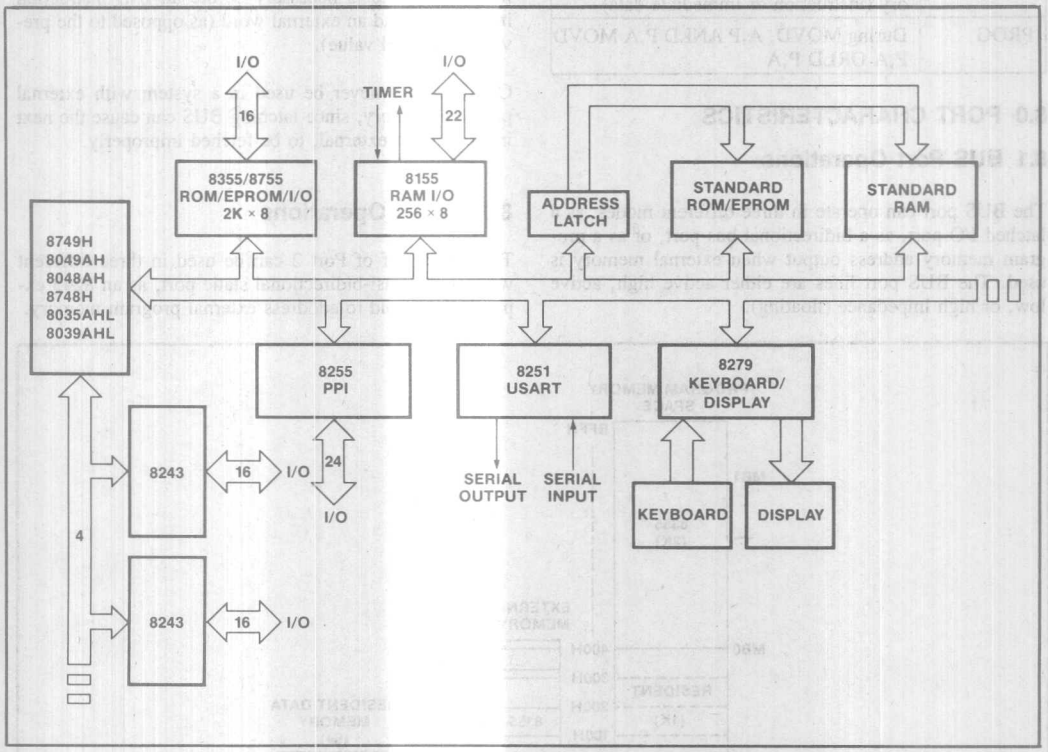


Figure 13. MCS® -48 Expansion Capability

MCS®-48 INSTRUCTION SET

1.0 INTRODUCTION

The MCS®-48 instruction set is extensive for a machine of its size and has been tailored to be straightforward and very efficient in its use of program memory. All instructions are either one or two bytes in length and over 80% are only one byte long. Also, all instructions execute in either one or two cycles and over 50% of all instructions execute in a single cycle. Double cycle instructions include all immediate instructions, and all I/O instructions.

The MCS-48 microcomputers have been designed to handle arithmetic operations efficiently in both binary and BCD as well as handle the single-bit operations required in control applications. Special instructions have also been included to simplify loop counters, table look-up routines, and N-way branch routines.

1.1 Data Transfers

As can be seen in Figure 1 the 8-bit accumulator is the central point for all data transfers within the 8048. Data can be transferred between the 8 registers of each working register bank and the accumulator directly, i.e., the source or destination register is specified by the instruction. The remaining locations of the internal RAM array are referred to as Data Memory and are addressed indirectly via an address stored in either R0 or R1 of the active register bank. R0 and R1 are also used to indirecly address external data memory when it is present. Transfers to and from internal RAM require one cycle, while transfers to external RAM require two. Constants stored in Program Memory can be loaded directly to the accumulator and to the 8 working registers. Data can also be transferred directly between the accumulator and the on-

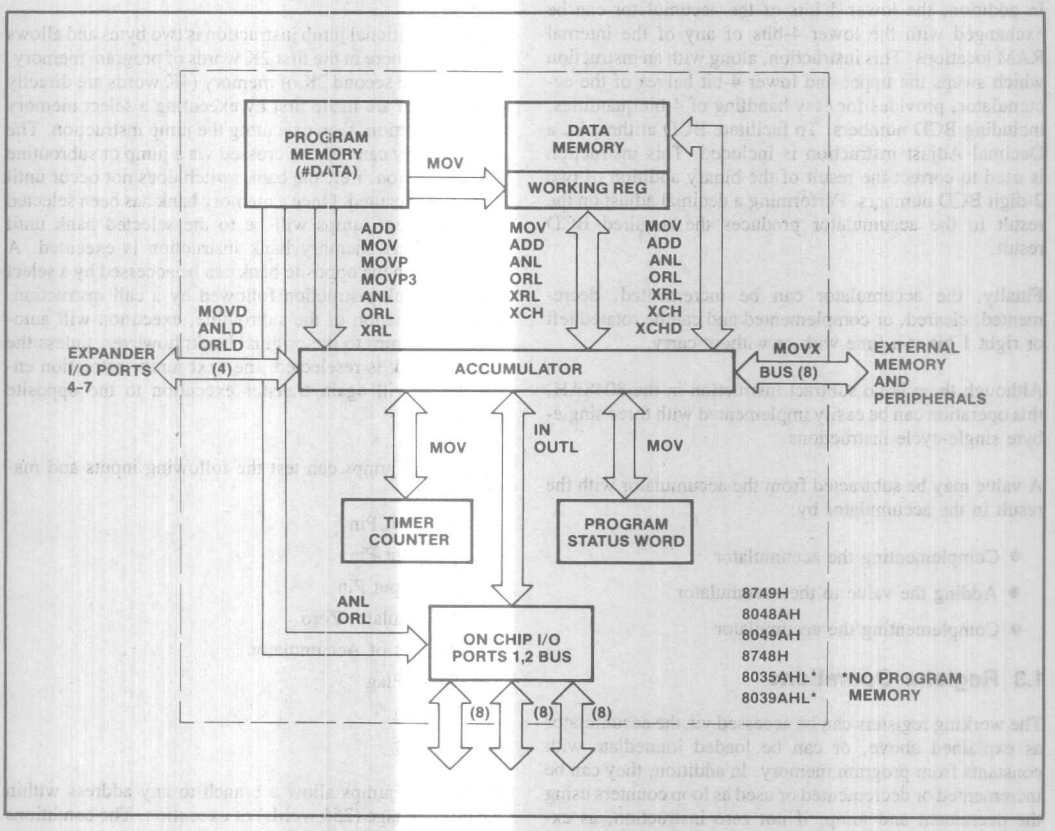


Figure 1. Data Transfer Instructions

board timer counter or the accumulator and the Program Status word (PSW). Writing to the PSW alters machine status accordingly and provides a means of restoring status after an interrupt or of altering the stack pointer if necessary.

1.2 Accumulator Operations

Immediate data, data memory, or the working registers can be added with or without carry to the accumulator. These sources can also be ANDed, ORed, or Exclusive ORed to the accumulator. Data may be moved to or from the accumulator and working registers or data memory. The two values can also be exchanged in a single operation.

In addition, the lower 4 bits of the accumulator can be exchanged with the lower 4-bits of any of the internal RAM locations. This instruction, along with an instruction which swaps the upper and lower 4-bit halves of the accumulator, provides for easy handling of 4-bit quantities, including BCD numbers. To facilitate BCD arithmetic, a Decimal Adjust instruction is included. This instruction is used to correct the result of the binary addition of two 2-digit BCD numbers. Performing a decimal adjust on the result in the accumulator produces the required BCD result.

Finally, the accumulator can be incremented, decremented, cleared, or complemented and can be rotated left or right 1 bit at a time with or without carry.

Although there is no subtract instruction in the 8048AH, this operation can be easily implemented with three single-byte single-cycle instructions.

A value may be subtracted from the accumulator with the result in the accumulator by:

- Complementing the accumulator
- Adding the value to the accumulator
- Complementing the accumulator

1.3 Register Operations

The working registers can be accessed via the accumulator as explained above, or can be loaded immediate with constants from program memory. In addition, they can be incremented or decremented or used as loop counters using the decrement and jump, if not zero instruction, as explained under branch instructions.

All Data Memory including working registers can be accessed with indirect instructions via R0 and R1 and can be incremented.

1.4 Flags

There are four user-accessible flags in the 8048AH: Carry, Auxiliary Carry, F0 and F1. Carry indicates overflow of the accumulator, and Auxiliary Carry is used to indiate overflow between BCD digits and is used during decimal-adjust operation. Both Carry and Auxiliary Carry are accessible as part of the program status word and are stored on the stack during subroutines. F0 and F1 are undedicated general-purpose flags to be used as the programmer desires. Both flags can be cleared or complemented and tested by conditional jump instructions. F0 is also accessible via the Program Status word and is stored on the stack with the carry flags.

1.5 Branch Instructions

The unconditional jump instruction is two bytes and allows jumps anywhere in the first 2K words of program memory. Jumps to the second 2K of memory (4K words are directly addressable) are made first by executing a select memory bank instruction, then executing the jump instruction. The 2K boundary can only be crossed via a jump or subroutine call instruction, i.e., the bank switch does not occur until a jump is executed. Once a memory bank has been selected all subsequent jumps will be to the selected bank until another select memory bank instruction is executed. A subroutine in the opposite bank can be accessed by a select memory bank instruction followed by a call instruction. Upon completion of the subroutine, execution will automatically return to the original bank; however, unless the original bank is reselected, the next jump instruction encountered will again transfer execution to the opposite bank.

Conditional jumps can test the following inputs and machine status:

- T0 Input Pin
- T1 Input Pin
- $\overline{\text{INT}}$ Input Pin
- Accumulator Zero
- Any bit of Accumulator
- Carry Flag
- F0 Flag
- F1 Flag

Conditional jumps allow a branch to any address within the current page (256 words) of execution. The conditions tested are the instantaneous values at the time the conditional jump is executed. For instance, the jump on accumulator zero instruction tests the accumulator itself, not an intermediate zero flag.

The decrement register and jump if not zero instruction combines a decrement and a branch instruction to create an instruction very useful in implementing a loop counter. This instruction can designate any one of the 8 working registers as a counter and can effect a branch to any address within the current page of execution.

A single-byte indirect jump instruction allows the program to be vectored to any one of several different locations based on the contents of the accumulator. The contents of the accumulator points to a location in program memory which contains the jump address. The 8-bit jump address refers to the current page of execution. This instruction could be used, for instance, to vector to any one of several routines based on an ASCII character which has been loaded in the accumulator. In this way ASCII key inputs can be used to initiate various routines.

1.6 Subroutines

Subroutines are entered by executing a call instruction. Calls can be made like unconditional jumps to any address in a 2K word bank, and jumps across the 2K boundary are executed in the same manner. Two separate return instructions determine whether or not status (upper 4-bits of PSW) is restored upon return from the subroutine.

The return and restore status instruction also signals the end of an interrupt service routine if one has been in progress.

1.7 Timer Instructions

The 8-bit on board timer/counter can be loaded or read via the accumulator while the counter is stopped or while counting. The counter can be started as a timer with an internal clock source or an event counter or timer with an external clock applied to the T1 input pin. The instruction executed determines which clock source is used. A single instruction stops the counter whether it is operating with an internal or an external clock source. In addition, two instructions allow the timer interrupt to be enabled or disabled.

1.8 Control Instructions

Two instructions allow the external interrupt source to be enabled or disabled. Interrupts are initially disabled and are automatically disabled while an interrupt service routines is in progress and re-enabled afterward.

There are four memory bank select instructions, two to designate the active working register bank and two to control program memory banks. The operation of the program memory bank switch is explained in Section 2.2 in the Expanded MCS-48 System chapter.

The working register bank switch instructions allow the programmer to immediately substitute a second 8-register working register bank for the one in use. This effectively provides 16 working registers or it can be used as a means of quickly saving the contents of the registers in response to an interrupt. The user has the option to switch or not to switch banks on interrupt. However, if the banks are switched, the original bank will be automatically restored upon execution of a return and restore status instruction at the end of the interrupt service routine.

A special instruction enables an internal clock, which is the XTAL frequency divided by three to be output on pin T0. This clock can be used as a general-purpose clock in the user's system. This instruction should be used only to initialize the system since the clock output can be disabled only by application of system reset.

1.9 Input/Output Instructions

Ports 1 and 2 are 8-bit static I/O ports which can be loaded to and from the accumulator. Outputs are statically latched but inputs are not latched and must be read while inputs are present. In addition, immediate data from program memory can be ANDed or ORed directly to Port 1 and Port 2 with the result remaining on the port. This allows "masks" stored in program memory to selectively set or reset individual bits of the I/O ports. Ports 1 and 2 are configured to allow input on a given pin by first writing a "1" out to the pin.

An 8-bit port called BUS can also be accessed via the accumulator and can have statically latched outputs as well. It too can have immediate data ANDed or ORed directly to its outputs, however, unlike ports 1 and 2, all eight lines of BUS must be treated as either input or output at any one time. In addition to being a static port, BUS can be used as a true synchronous bi-directional port using the Move External instructions used to access external data memory. When these instructions are executed, a corresponding $\overline{\text{READ}}$ or $\overline{\text{WRITE}}$ pulse is generated and data is valid only at that time. When data is not being transferred, BUS is in a high impedance state. Note that the OUTL, ANL, and the ORL instructions for the BUS are for use with internal program memory only.

The basic three on-board I/O ports can be expanded via a 4-bit expander bus using half of port 2. I/O expander devices on this bus consist of four 4-bit ports which are addressed as ports 4 through 7. These ports have their own AND and OR instructions like the on-board ports as well as move instructions to transfer data in or out. The expander AND and OR instructions, however, combine the contents of accumulator with the selected port rather than immediate data as is done with the on-board ports.

I/O devices can also be added externally using the BUS port as the expansion bus. In this case the I/O ports become "memory mapped", i.e., they are addressed in the same way as external data memory and exist in the external data memory address space addressed by pointer register R0 or R1.

2.0 INSTRUCTION SET DESCRIPTION

The following pages describe the MCS®-48 instruction set in detail. The instruction set is first summarized with instructions grouped functionally. This summary page is followed by a detailed description listed alphabetically by mnemonic opcode.

The alphabetical listing includes the following information.

- Mnemonic
- Machine Code
- Verbal Description
- Symbolic Description
- Assembly Language Example

The machine code is represented with the most significant bit (7) to the left and two byte instructions are represented with the first byte on the left. The assembly language examples are formulated as follows:

Arbitrary

Label: Mnemonic, Operand;

Descriptive Comment

Instruction Set Summary

Mnemonic	Description	Bytes	Cycle
Accumulator			
ADD A, R	Add register to A	1	1
ADD A, @R	Add data memory to A	1	1
ADD A, # data	Add immediate to A	2	2
ADDC A, R	Add register with carry	1	1
ADDC A, @R	Add data memory with carry	1	1
ADDC A, # data	Add immediate with carry	2	2
ANL A, R	And register to A	1	1
ANL A, @R	And data memory to A	1	1
ANL A, # data	And immediate to A	2	2
ORL A, R	Or register to A	1	1
ORL A @R	Or data memory to A	1	1
ORL A, # data	Or immediate to A	2	2
XRL A, R	Exclusive Or register to A	1	1
XRL A, @R	Exclusive or data memory to A	1	1
XRL, A, # data	Exclusive or immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
Input/Output			
IN A, P	Input port to A	1	2
OUTL P, A	Output A to port	1	2
ANL P, # data	And immediate to port	2	2
ORL P, # data	Or immediate to port	2	2
*INS A, BUS	Input BUS to A	1	2
*OUTL BUS, A	Output A to BUS	1	2
*ANL BUS, # data	And immediate to BUS	2	2
*ORL BUS, # data	Or immediate to BUS	2	2
MOVD A, P	Input Expander port to A	1	2
MOVD P, A	Output A to Expander port	1	2
ANLD P, A	And A to Expander port	1	2
ORLD P, A	Or A to Expander port	1	2

Mnemonic	Description	Bytes	Cycles
Registers			
INC R	Increment register	1	1
INC @R	Increment data memory	1	1
DEC R	Decrement register	1	1
Branch			
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R, addr	Decrement register and jump	2	2
JC addr	Jump on carry = 1	2	2
JNC addr	Jump on carry = 0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0 = 1	2	2
JNT0 addr	Jump on T0 = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 = 1	2	2
JF1 addr	Jump on F1 = 1	2	2
JTF addr	Jump on timer flag = 1	2	2
JNI addr	Jump on INT = 0	2	2
JBb addr	Jump on Accumulator Bit	2	2
Subroutine			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
Flags			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear Flag 1	1	1
CPL F1	Complement Flag 1	1	1
Data Moves			
MOV A, R	Move register to A	1	1
MOV A, @R	Move data memory to A	1	1
MOV A, # data	Move immediate to A	2	2
MOV R, A	Move A to register	1	1
MOV @R, A	Move A to data memory	1	1
MOV R, # data	Move immediate to register	2	2
MOV @R, # data	Move immediate to data memory	2	2
MOV A, PSW	Move PSW to A	1	1
MOV PSW, A	Move A to PSW	1	1

Mnemonics copyright Intel Corporation 1983.

*For use with internal memory only.

Mnemonic	Description	Bytes	Cycle
Data Moves (Cont'd)			
XCH A, R	Exchange A and register	1	1
XCH A, @R	Exchange A and data memory	1	1
XCHD A, @R	Exchange nibble of A and register	1	1
MOVX A, @R	Move external data memory to A	1	2
MOVX @R, A	Move A to external data memory	1	2
MOVP A, @A	Move to A from current page	1	2
MOVP3 A, @A	Move to A from Page 3	1	2
Timer/Counter			
MOV A, T	Read Timer/Counter	1	1
MOV T, A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	Start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1

Mnemonic	Description	Bytes	Cycle
Control			
EN I	Enable external Interrupt	1	1
DIS I	Disable external Interrupt	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
SEL MB0	Select memory bank 0	1	1
SEL MB1	Select memory bank 1	1	1
ENT0 CLK	Enable clock output on T0	1	1
NOP	No Operation	1	1

Mnemonics copyright Intel Corporation 1983.

MCS®-48 INSTRUCTION SET
Symbols and Abbreviations Used

A	Accumulator
AC	Auxiliary Carry
addr	12-Bit Program Memory Address
Bb	Bit Designator (b = 0–7)
BS	Bank Switch
BUS	BUS Port
C	Carry
CLK	Clock
CNT	Event Counter
CRR	Conversion Result Register
D	Mnemonic for 4-Bit Digit (Nibble)
data	8-Bit Number or Expression
DBF	Memory Bank Flip-Flop
F0, F1	Flag 0, Flag 1
I	Interrupt
P	Mnemonic for "in-page" Operation
PC	Program Counter
Pp	Port Designator (p = 1, 2 or 4–7)
PSW	Program Status Word
Ri	Data memory Pointer (i = 0, or 1)
Rr	Register Designator (r = 0–7)
SP	Stack Pointer
T	Timer
TF	Timer Flag
T0, T1	Test 0, Test 1
X	Mnemonic for External RAM
#	Immediate Data Prefix
@	Indirect Address Prefix
$	Current Value of Program Counter
(X)	Contents of X
((X))	Contents of Location Addressed by X
←	Is Replaced by

Mnemonics copyright Intel Corporation 1983.

ADD A,R_r Add Register Contents to Accumulator

Encoding: | 0 1 1 0 | 1 r r r | 68H-6FH

Description: The contents of register 'r' are added to the accumulator. Carry is affected.

Operation: (A) ← (A) + (Rr) r = 0–7

Example:
```
ADDREG: ADD A,R6      ;ADD REG 6 CONTENTS
                      ;TO ACC
```

ADD A,@R_i Add Data Memory Contents to Accumulator

Encoding; | 0 1 1 0 | 0 0 0 i | 60H-61H

Description: The contents of the resident data memory location addressed by register 'i' bits 0–5** are added to the accumulator. Carry is affected.

Operation: (A) ← (A) + ((Ri)) i = 0–1

Example:
```
ADDM: MOV R0, #01FH   ;MOVE '1F' HEX TO REG 0
      ADD A, @R0      ;ADD VALUE OF LOCATION
                      ;31 TO ACC
```

ADD A,#data Add Immediate Data to Accumulator

Encoding: | 0 0 0 0 | 0 0 1 1 | d_7 d_6 d_5 d_4 | d_3 d_2 d_1 d_0 | 03H

Description: This is a 2-cycle instruction. The specified data is added to the accumulator. Carry is affected.

Operation: (A) ← (A) + data

Example:
```
ADDID: ADD A,#ADDER:  ;ADD VALUE OF SYMBOL
                      ;ADDER' TO ACC
```

ADDC A,R_r Add Carry and Register Contents to Accumulator

Encoding: | 0 1 1 1 | 1 r r r | 78H-7FH

Description: The content of the carry bit is added to accumulator location 0 and the carry bit cleared. The contents of register 'r' are then added to the accumulator. Carry is affected.

Operation: (A) ← (A) + (Rr) + (C) r = 0–7

Example:
```
ADDRGC: ADDC A,R4     ;ADD CARRY AND REG 4
                      ;CONTENTS TO ACC
```

** 0–5 in 8048AH/8748H
0–6 in 8049AH/8749H
0–7 in 8050AH

ADDC A,@R_i Add Carry and Data Memory Contents to Accumulator

Encoding: | 0 1 1 1 | 0 0 0 i | 70H-71H

Description: The content of the carry bit is added to accumulator location 0 and the carry bit cleared. Then the contents of the resident data memory location addressed by register 'i' bits 0–5** are added to the accumulator. Carry is affected.

Operation: (A) ← (A) + ((Ri)) + (C) i = 0-1

Example:
```
ADDMC:  MOV R1,#40      ;MOVE '40' DEC TO REG 1
        ADDC A,@R1      ;ADD CARRY AND LOCATION 40
                        ;CONTENTS TO ACC
```

ADDC A,@data Add Carry and Immediate Data to Accumulator

Encoding: | 0 0 0 1 | 0 0 1 1 | d_7 d_6 d_5 d_4 | d_3 d_2 d_1 d_0 | 13H

Description: This is a 2-cycle instruction. The content of the carry bit is added to accumulator location 0 and the carry bit cleared. Then the specified data is added to the accumulator. Carry is affected.

Operation: (A) ← (A) + data + (C)

Example:
```
ADDC A,#225             ;ADD CARRY AND '225' DEC
                        ;TO ACC
```

ANL A,R_r Logical AND Accumulator with Register Mask

Encoding: | 0 1 0 1 | 1 r r r | 58H-5FH

Description: Data in the accumulator is logically ANDed with the mask contained in working register 'r'.

Operation: (A) ← (A) AND (Rr) r = 0-7

Example:
```
ANDREG: ANL A,R3        ;'AND' ACC CONTENTS WITH MASK
                        ;IN REG 3
```

ANL A,@R_i Logical AND Accumulator with memory Mask

Encoding: | 0 1 0 1 | 0 0 0 i | 50H-51H

Description: Data in the accumulator is logically ANDed with the mask contained in the data memory location referenced by register 'i' bits 0–5**.

Operation: (A) ← (A) AND ((Ri)) i = 0-1

Example:
```
ANDDM:  MOV R0,#03FH    ;MOVE '3F' HEX TO REG 0
        ANL A, @R0      ;'AND' ACC CONTENTS WITH
                        ;MASK IN LOCATION 63
```

** 0–5 in 8048AH/8748H
0–6 in 8049AH/8749H
0–7 in 8050AH

ANL A,#data Logical AND Accumulator with Immediate Mask

Encoding: | 0 1 0 1 | 0 0 1 1 | d_7 d_6 d_5 d_4 | d_3 d_2 d_1 d_0 | 53H

Description: This is a 2-cycle instruction. Data in the accumulator is logically ANDed with an immediately-specified mask.

Operation: (A) ← (A) AND data

Examples:
```
ANDID: ANL A,#0AFH      ;'AND' ACC CONTENTS
                        ;WITH MASK 10101111
       ANL A,#3 + X/Y   ;'AND' ACC CONTENTS
                        ;WITH VALUE OF EXP
                        ;'3 + XY/Y'
```

ANL BUS,#data* Logical AND BUS with Immediate Mask

Encoding: | 1 0 0 1 | 1 0 0 0 | d_7 d_6 d_5 d_4 | d_3 d_2 d_1 d_0 | 98H

Description: This is a 2-cycle instruction. Data on the BUS port is logically ANDed with an immediately-specified mask. This instruction assumes prior specification of an 'OUTL BUS, A' instruction.

Operation: (BUS) ← (BUS) AND data

Example:
```
ANDBUS: ANL BUS,#MASK   ;'AND' BUS CONTENTS
                        ;WITH MASK EQUAL VALUE
                        ;OF SYMBOL 'MASK'
```

ANL Pp,#data Logical AND Port 1-2 with Immediate Mask

Encoding: | 1 0 0 1 | 1 0 p p | d_7 d_6 d_5 d_4 | d_3 d_2 d_1 d_0 | 99H-9AH

Description: This is a 2-cycle instruction. Data on port 'p' is logically ANDed with an immediately-specified mask.

Operation: (Pp) ← (Pp) AND DATA p = 1-2

Example:
```
ANDP2: ANL P2,#0F0H     ;'AND' PORT 2 CONTENTS
                        ;WITH MASK 'F0' HEX
                        ;(CLEAR P20-23)
```

* For use with internal program memory ONLY.

ANLD Pp,A Logical AND Port 4-7 with Accumulator Mask

Encoding: | 1 0 0 1 | 1 1 p p | 9CH-9FH

Description: This is a 2-cycle instruction. Data on port 'p' is logically ANDed with the digit mask contained in accumulator bits 0-3.

Operation: (Pp) ← (Pp) AND (A0-3) p = 4-7

Note: The mapping of port 'p' to opcode bits 0-1 is as follows:

1 0	Port
0 0	4
0 1	5
1 0	6
1 1	7

Example:
```
ANDP4:  ANLD P4,A        ;'AND' PORT 4 CONTENTS
                         ;WITH ACC BITS 0-3
```

CALL address Subroutine Call

Encoding: | a_{10} a_9 a_8 1 | 0 1 0 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 |

Page	Hex Op Code
0	14
1	34
2	54
3	74
4	94
5	B4
6	D4
7	F4

Description: This is a 2-cycle instruction. The program counter and PSW bits 4-7 are saved in the stack. The stack pointer (PSW bits 0-2) is updated. Program control is then passed to the location specified by 'address'. PC bit 11 is determined by the most recent SEL MB instruction.

A CALL cannot begin in locations 2046-2047 or 4094-4095. Execution continues at the instruction following the CALL upon return from the subroutine.

Operation:
$((SP)) \leftarrow (PC), (PSW_{4-7})$
$(SP) \leftarrow (SP) + 1$
$(PC_{8-10}) \leftarrow (addr_{8-10})$
$(PC_{0-7}) \leftarrow (addr_{0-7})$
$(PC_{11}) \leftarrow DBF$

Example: Add three groups of two numbers. Put subtotals in locations 50, 51 and total in location 52.

```
                MOV R0,#50      ;MOVE '50' DEC TO ADDRESS
                                ;REG 0
BEGADD:         MOV A,R1        ;MOVE CONTENTS OF REG 1
                                ;TO ACC
                ADD A,R2        ;ADD REG 2 TO ACC
                CALL SUBTOT     ;CALL SUBROUTINE 'SUBTOT'
                ADDC A R3       ;ADD REG 3 TO ACC
                ADDC A,R4       ;ADD REG 4 TO ACC
                CALL SUBTOT     ;CALL SUBROUTINE 'SUBTOT'
                ADDC A,R5       ;ADD REG 5 TO ACC
                ADDC A,R6       ;ADD REG 6 TO ACC
                CALL SUBTOT     ;CALL SUBROUTINE 'SUBTOT'
SUBTOT:         MOV @R0,A       ;MOVE CONTENTS OF ACC TO
                                ;LOCATION ADDRESSED BY
                                ;REG 0
                INC R0          ;INCREMENT REG 0
                RET             ;RETURN TO MAIN PROGRAM
```

CLR A Clear Accumulator

Encoding: | 0 0 1 0 | 0 1 1 1 | 27H

Description: The contents of the accumulator are cleared to zero.

Operation: A ← 0

CLR C Clear Carry Bit

Encoding: | 1 0 0 1 | 0 1 1 1 | 97H

Description: During normal program execution, the carry bit can be set to one by the ADD, ADDC, RLC, CPL C, RRC, and DAA insructions. This instruction resets the carry bit to zero.

Operation: C ← 0

CLR F1 Clear Flag 1

Encoding: | 1 0 1 0 | 0 1 0 1 | A5H

Description: Flag 1 is cleared to zero.

Operation: (F1) ← 0

CLR F0 Clear Flag 0

Encoding: 1 0 0 0 | 0 1 0 1 — 85H

Description: Flag 0 is cleared to zero.

Operation: (F0) ← 0

CPL A Complement Accumulator

Encoding: 0 0 1 1 | 0 1 1 1 — 37H

Description: The contents of the accumulator are complemented. This is strictly a one's complement. Each one is changed to zero and vice-versa.

Operation: (A) ← NOT (A)

Example: Assume accumulator contains 01101010.

```
CPLA: CPL A          ;ACC CONTENTS ARE COMPLE-
                     ;MENTED TO 10010101
```

CPL C Complement Carry Bit

Encoding: 1 0 1 0 | 0 1 1 1 — A7H

Description: The setting of the carry bit is complemented; one is changed to zero, and zero is changed to one.

Operation: (C) ← NOT (C)

Example: Set C to one; current setting is unknown.

```
CTO1: CLR C          ;C IS CLEARED TO ZERO
      CPL C          ;C IS SET TO ONE
```

CPL F0 Complement Flag 0

Encoding: 1 0 0 1 | 0 1 0 1 — 95H

Description: The setting of flag 0 is complemented; one is changed to zero, and zero is changed to one.

Operation: F0 ← NOT (F0)

CPL F1 Complement Flag 1

Encoding: 1 0 1 1 | 0 1 0 1 — B5H

Description: The setting of flag 1 is complemented; one is changed to zero, and zero is changed to one.

Operation: (F1) ← NOT (F1)

DA A Decimal Adjust Accumulator

Encoding: | 0 1 0 1 | 0 1 1 1 | 57H

Description: The 8-bit accumulator value is adjusted to form two 4-bit Binary Coded Decimal (BCD) digits following the binary addition of BCD numbers. The carry bit C is affected. If the contents of bits 0–3 are greater than nine, or if AC is one, the accumulator is incremented by six.

The four high-order bits are then checked. If bits 4–7 exceed nine, or if C is one, these bits are increased by six. If an overflow occurs, C is set to one.

Example: Assume accumulator contains 10011011.

```
DA A                        ;ACC Adjusted to 00000001
                            ;WITH C SET

C AC 7     4 3     0
0  0 1 0 0 1 1 0 1 1
     0 0 0 0 0 1 1 0        ADD SIX TO BITS 0-7
0  1 1 0 1 0 0 0 0 1
       0 1 1 0              ADD SIX TO BITS 4-7
1  0 0 0 0 0 0 0 0 1        OVERFLOW TO C
```

DEC A Decrement Accumulator

Encoding: | 0 0 0 0 | 0 1 1 1 | 07H

Description: The contents of the accumulator are decremented by one. The carry flag is not affected.

Operation: (A) ← (A) -1

Example: Decrement contents of external data memory location 63.

```
MOV R0,#3FH                 ;MOVE '3F' HEX TO REG 0
MOVX A, @R0                 ;MOVE CONTENTS OF
                            ;LOCATION 63 TO ACC
DEC A                       ;DECREMENT ACC
MOVX @R0,A                  ;MOVE CONTENTS OF ACC TO
                            ;LOCATION 63 IN EXPANDED
                            ;MEMORY
```

DEC Rr Decrement Register

Encoding: | 1 1 0 0 | 1 r r r | C8H-CFH

Description: The contents of working register 'r' are decremented by one.

Operation: (Rr) ← (Rr) -1 r = 0-7

Example:

```
DECR1: DEC R1               ;DECREMENT CONTENTS OF REG 1
```

DIS I External Interrupt

Encoding: | 0 0 0 1 | 0 1 0 1 | 15H

Description: External interrupts are disabled. A low signal on the interrupt input pin has no effect.

DIS TCNTI Disable Timer/Counter Interrupt

Encoding: | 0 0 1 1 | 0 1 0 1 | 35H

Description: Timer/counter interrupts are disabled. Any pending timer interrupt request is cleared. The interrupt sequence is not initiated by an overflow, but the timer flag is set and time accumulation continues.

DJNZ R_r, address Decrement Register and Test

Encoding: | 1 1 1 0 | 1 r r r | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | E8H–EFH

Description: This is a 2-cycle instruction. Register 'r' is decremented, then tested for zero. If the register contains all zeros, program control falls through to the next instruction. If the register contents are not zero, control jumps to the specified 'address'.

The address in this case must evaluate to 8-bits, that is, the jump must be to a location within the current 256-location page.

Example: (Rr) ← (Rr) –1 r = 0–7
If Rr not 0
(PC_{0-7}) ← addr
Note: A 12-bit address specification does not cause an error if the DJNZ instruction and the jump target are on the same page. If the DJNZ instruction begins in location 255 of a page, it must jump to a target address on the following page.

Example: Increment values in data memory locations 50–54.

```
        MOV R0,#50          ;MOVE '50' DEC TO ADDRESS
                            ;REG 0
        MOV R3,#5           ;MOVE '5' DEC TO COUNTER
                            ;REG 3
INCRT:  INC @R0             ;INCREMENT CONTENTS OF
                            ;LOCATION ADDRESSED BY
                            ;REG 0
        INC R0              ;INCREMENT ADDRESS IN REG 0
        DJNZ R3, INCRT      ;DECREMENT REG 3 — JUMP TO
                            ;'INCRT' IF REG 3 NONZERO
        NEXT —              ;'NEXT' ROUTINE EXECUTED
                            ;IF R3 IS ZERO
```

EN I Enable External Interrupt

Encoding: | 0 0 0 0 | 0 1 0 1 | 05H

Description: External interrupts are enabled. A low signal on the interrupt input pin initiates the interrupt sequence.

EN TCNTI Enable Timer/Counter Interrupt

Encoding: | 0 0 1 0 | 0 1 0 1 | 25H

Description: Timer/counter interrupts are enabled. An overflow of the timer/counter initiates the interrupt sequence.

ENT0 CLK Enable Clock Output

Encoding: | 0 1 1 1 | 0 1 0 1 | 75H

Description: The test 0 pin is enabled to act as the clock output. This function is disabled by a system reset.

Example:
```
EMTST0:  ENT0 CLK          ;ENABLE T0 AS CLOCK OUTPUT
```

IN A,Pp Input Port or Data to Accumulator

Encoding: | 0 0 0 0 | 1 0 p p | 09H-0AH

Description: This is a 2-cycle instruction. Data present on port 'p' is transferred (read) to the accumulator.

Operation: (A) ← (Pp)

```
                    p = 1-2
INP12:  IN A,P1     ;INPUT PORT 1 CONTENTS TO ACC
        MOV R6,A    ;MOVE ACC CONTENTS TO REG 6
        IN A,P2     ;INPUT PORT 2 CONTENTS TO ACC
        MOV R7,A    ;MOVE ACC CONTENTS TO REG 7
```

INC A Increment Accumulator

Encoding: | 0 0 0 1 | 0 1 1 1 | 17H

Description: The contents of the accumulator are incremented by one. Carry is not affected.

Operation. (A) ← (A) +1

Example: Increment contents of location 100 in external data memory.

```
INCA:  MOV R0,#100      ;MOVE '100' DEC TO ADDRESS REG 0
       MOVX A,@R0       ;MOVE CONTENTS OF LOCATION
                        ;100 TO ACC
       INC A            ;INCREMENT A
       MOVX @R0,A       ;MOVE ACC CONTENTS TO
                        ;LOCATION 101
```

INC R_r Increment Register

Encoding: | 0 0 0 1 | 1 r r r | 18H–1FH

Description: The contents of working register 'r' are incremented by one.

Operation: (Rr) ← (Rr) + 1 r = 0–7

```
Example: INCR0:  INC R0          ;INCREMENT CONTENTS OF REG 0
```

INC @R_i Increment Data Memory Location

Encoding: | 0 0 0 1 | 0 0 0 i | 10H–11H

Description: The contents of the resident data memory location addressed by register 'i' bits 0–5** are incremented by one.

Operation: ((Ri)) ← ((Ri)) + 1 i = 0–1

```
Example: INCDM:  MOV R1,#03FH    ;MOVE ONES TO REG 1
                 INC @R1         ;INCREMENT LOCATION 63
```

INS A,BUS* Strobed Input of BUS Data to Accumulator

Encoding: | 0 0 0 0 | 1 0 0 0 | 08H

Description: This is a 2-cycle instruction. Data present on the BUS port is transferred (read) to the accumulator when the RD pulse is dropped. (Refer to section on programming memory expansion for details.)

Operation: (A) ← (BUS)

```
Example: INPBUS:  INS A,BUS      ;INPUT BUS CONTENTS TO ACC
```

* For use with internal program memory ONLY.

** 0–5 in 8048AH/8748H
0–6 in 8049AH/8749H
0–7 in 8050AH

JBb address Jump If Accumulator Bit Is Set

Encoding: | b_2 b_1 b_0 1 | 0 0 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 |

Accumulator Bit	Hex Op Code
0	12
1	32
2	52
3	72
4	92
5	B2
6	D2
7	F2

Description: This is a 2-cycle instruction. Control passes to the specified address if accumulator bit 'b' is set to one.

Operation:

b = 0–7
(PC_{0-7}) ← addr — If Bb = 1
(PC) = (PC) + 2 — If Bb = 0

Example: JB4IS1: JB4 NEXT ;JUMP TO 'NEXT' ROUTINE
;IF ACC BIT 4 = 1

JC address Jump If Carry Is Set

Encoding: | 1 1 1 1 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | F6H

Description: This is a 2-cycle instruction. Control passes to the specified address if the carry bit is set to one.

Operation:

(PC_{0-7}) ← addr — If C = 1
(PC) = (PC) + 2 — If C = 0

Example: JC1: JC OVFLOW ;JUMP TO 'OVFLOW' ROUTINE
;IF C = 1

JF0 address Jump If Flag 0 Is Set

Encoding: | 1 0 1 1 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | B6H

Description: This is a 2-cycle instruction. Control passes to the specified address if flag 0 is set to one.

Operation:

(PC_{0-7}) ← addr — If F0 = 1
(PC) =(PC) + 2 — If F0 = 0

Example: JF0IS1: JF0 TOTAL ;JUMP TO 'TOTAL' ROUTINE IF F0 = 1

JF1 address Jump If Flag 1 Is Set

Encoding: | 0 1 1 1 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | 76H

Description: This is a 2-cycle instruction. Control passes to the specified address if flag 1 is set to one.

Operation:
$(PC_{0-7}) \leftarrow$ addr If F1 = 1
(PC) =(PC + 2) If F1 = 0

Example:
```
JF1IS1: JF1 FILBUF      ;JUMP TO 'FILBUF'
                        ;ROUTINE IF F1 = 1
```

JMP address Direct Jump within 2K Block

Encoding: | a_{10} a_9 a_8 0 | 0 1 0 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 |

Page	Hex Op Code
0	04
1	24
2	44
3	64
4	84
5	A4
6	C4
7	E4

Description: This is a 2-cycle instruction. Bits 0–10 of the program counter are replaced with the directly-specified address. The setting of PC bit 11 is determined by the most recent SELECT MB instruction.

Operation:
$(PC_{8-10}) \leftarrow$ addr 8–10
$(PC_{0-7}) \leftarrow$ addr 0–7
$(PC_{11}) \leftarrow$ DBF

Example:
```
JMP SUBTOT      ;JUMP TO SUBROUTINE 'SUBTOT'
JMP $-6         ;JUMP TO INSTRUCTION SIX
                ;LOCATIONS BEFORE CURRENT
                ;LOCATION
JMP 2FH         ;JUMP TO ADDRESS '2F' HEX
```

JMPP @A Indirect Jump within Page

Encoding: | 1 0 1 1 | 0 0 1 1 | B3H

Description: This is a 2-cycle insruction. The contents of the program memory location pointed to by the accumulator are substituted for the 'page' portion of the program counter (PC bits 0–7).

Operation: $(PC_{0-7}) \leftarrow ((A))$

Example: Assume accumulator contains 0FH.

```
JMPPAG: JMPP @A          ;JUMP TO ADDRESS STORED IN
                         ;LOCATION 15 IN CURRENT PAGE
```

JNC address Jump If Carry Is Not Set

Encoding: | 1 1 1 0 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | E6H

Description: This is a 2-cycle instruction. Control passes to the specified address if the carry bit is not set, that is, equals zero.

Operation: $(PC_{0-7}) \leftarrow$ addr — If C = 0
(PC) = (PC) + 2 — If C = 1

Example:

```
JC0: JNC NOVFLO          ;JUMP TO 'NOVFLO' ROUTINE
                         ;IF C = 0
```

JNI address Jump If Interrupt Input Is Low

Encoding: | 1 0 0 0 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | 86H

Description: This is a 2-cycle instruction. Control passes to the specified address if the interrupt input signal is low (= 0), that is, an external interrupt has been signaled. (This signal initiates an interrupt service sequence if the external interrupt is enabled.)

Operation: $(PC_{0-7}) \leftarrow$ addr — If I = 0
(PC) = (PC) + 2 — If I = 1

Example:

```
LOC 3: JNI EXTINT        ;JUMP TO 'EXTINT' ROUTINE
                         ;IF I = 0
```

JNT0 address Jump If Test 0 is Low

Encoding: | 0 0 1 0 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | 26H

Description: This is a 2-cycle instruction. Control passes to the specified address, if the test 0 signal is low.

Operation: $(PC_{0-7}) \leftarrow$ addr — If T0 = 0
(PC) = (PC) + 2 — If T0 = 1

Example:

```
JT0LOW: JNT0 60          ;JUMP TO LOCATION 60 DEC
                         ;IF T0 = 0
```

JNT1 address Jump If Test 1 Is Low

Encoding: | 0 1 0 0 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | 46H

Description: This is a 2-cycle instruction. Control passes to the specified address, if the test 1 signal is low.

Operation: $(PC_{0-7}) \leftarrow$ addr — If T1 = 0
(PC) = (PC) + 2 — If T1 = 1

JNZ Address Jump If Accumulator Is Not Zero

Encoding: | 1 0 0 1 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | 96H

Description: This is a 2-cycle instruction. Control passes to the specified address if the accumulator contents are nonzero at the time this instruction is executed.

Operation: $(PC_{0-7}) \leftarrow$ addr — If $A \neq 0$
(PC) = (PC) + 2 — If A = 0

Example: JACCN0: JNZ 0ABH ;JUMP TO LOCATION 'AB' HEX
;IF ACC VALUE IS NONZERO

JTF address Jump If Timer Flag Is Set

Encoding: | 0 0 0 1 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | 16H

Description: This is a 2-cycle instruction. Control passes to the specified address if the timer flag is set to one, that is, the timer/counter register has overflowed. Testing the timer flag resets it to zero. (This overflow initiates an interrupt service sequence if the timer-overflow interrupt is enabled.)

Operation: $(PC_{0-7}) \leftarrow$ addr — If TF = 1
(PC) = (PC) + 2 — If TF = 0

Example: JTF1: JTF TIMER ;JUMP TO 'TIMER' ROUTINE
;IF TF = 1

JT0 address Jump If Test 0 Is High

Encoding: | 0 0 1 1 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | 36H

Description: This is a 2-cycle instruction. Control passes to the specified address if the test 0 signal is high (= 1).

Operation: $(PC_{0-7}) \leftarrow$ addr — If T0 = 1
(PC) = (PC) + 2 — If T0 = 0

Example: JT0HI: JT0 53 ;JUMP TO LOCATION 53 DEC
;IF T0 = 1

JT1 address Jump If Test 1 Is High

Encoding: | 0 1 0 1 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | 56H

Description: This is a 2-cycle instruction. Control passes to the specified address if the test 1 signal is high (= 1).

Operation: $(PC_{0-7}) \leftarrow$ addr — If T1 = 1
(PC) = (PC) + 2 — If T1 = 0

Example:
```
JT1HI: JT1 COUNT        ;JUMP TO 'COUNT' ROUTINE
                        ;IF T1 = 1
```

JZ address Jump If Accumulator Is Zero

Encoding: | 1 1 0 0 | 0 1 1 0 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | C6H

Description: This is a 2-cycle instruction. Control passes to the specified address if the accumulator contains all zeros at the time this instruction is executed.

Operation: $(PC_{0-7}) \leftarrow$ addr — If A = 0
(PC) = (PC) + 2 — If A $\neq$ 1

Example:
```
JACCO: JZ 0A3H          ;JUMP TO LOCATION 'A3' HEX
                        ;IF ACC VALUE IS ZERO
```

MOV A,#data Move Immediate Data to Accumulator

Encoding: | 0 0 1 0 | 0 0 1 1 | a_7 a_6 a_5 a_4 | a_3 a_2 a_1 a_0 | 23H

Description: This is a 2-cycle instruction. The 8-bit value specified by 'data' is loaded in the accumulator.

Operation: (A) $\leftarrow$ data

Example:
```
MOV A,#0A3H             ;MOVE 'A3' HEX TO ACC
```

MOV A,PSW Move PSW Contents to Accumulator

Encoding: | 1 1 0 0 | 0 1 1 1 | C7H

Description: The contents of the program status word are moved to the accumulator.

Operation: (A) $\leftarrow$ (PSW)

Example: Jump to 'RB1SET' routine if PSW bank switch, bit 4, is set.
```
BSCHK:  MOV A,PSW       ;MOVE PSW CONTENTS TO ACC
        JB4 RB1SET      ;JUMP TO 'RB1SET' IF ACC BIT 4 = 1
```

MOV A,R_r Move Register Contents to Accumulator

Encoding: | 1 1 1 1 | 1 r r r | F8H–FFH

Description: 8-bits of data are removed from working register 'r' into the accumulator.

Operation: (A) ← (Rr) r = 0–7

Example:
```
MAR: MOV A,R3          ;MOVE CONTENTS OF REG 3 TO ACC
```

MOV A,@R_i Move Data Memory Contents to Accumulator

Encoding | 1 1 1 1 | 0 0 0 i | F0H–F1H

Description: The contents of the resident data memory location addressed by bits 0–5** of register 'i' are moved to the accumulator. Register 'i' contents are unaffected.

Operation: (A) ← ((Ri)) i = 0–1

Example: Assume R1 contains 00110110.
```
MADM: MOV A,@R1        ;MOVE CONTENTS OF DATA MEM
                       ;LOCATION 54 TO ACC
```

MOV A,T Move Timer/Counter Contents to Accumulator

Encoding: | 0 1 0 0 | 0 0 1 0 | 42H

Description: The contents of the timer/event-counter register are moved to the accumulator.

Operation: (A) ← (T)

Example: Jump to "EXIT" routine when timer reaches '64', that is, when bit 6 set—assuming initialization 64,
```
TIMCHK: MOV A,T        ;MOVE TIMER CONTENTS TO ACC
        JB6 EXIT       ;JUMP TO 'EXIT' IF ACC BIT 6 = 1
```

MOV PSW,A Move Accumulator Contents to PSW

Encoding: | 1 1 0 1 | 0 1 1 1 | D7H

Description: The contents of the accumulator are moved into the progam status word. All condition bits and the stack pointer are affected by this move.

Operation: (PSW) ← (A)

Example: Move up stack pointer by two memory locations, that is, increment the pointer by one.
```
INCPTR: MOV A,PSW      ;MOVE PSW CONTENTS TO ACC
        INC A          ;INCREMENT ACC BY ONE
        MOV PSW,A      ;MOVE ACC CONTENTS TO PSW
```

** 0–5 in 8048AH/8748H
0–6 in 8049AH/8749H
0–7 in 8050AH

MOV R_r,A Move Accumulator Contents to Register

Encoding: | 1 0 1 0 | 1 r r r | A8H–AFH

Description: The contents of the accumulator are moved to register 'r'.

Operation: (Rr) ← (A) r = 0–7

Example:
```
MRA: MOV R0,A          ;MOVE CONTENTS OF ACC TO REG 0
```

MOV R_r,#data Move Immediate Data to Register

Encoding: | 1 0 1 1 | 1 r_2 r_1 r_0 | | d_7 d_6 d_5 d_4 | d_3 d_2 d_1 d_0 | B8H–BFH

Description: This is a 2-cycle instruction. The 8-bit value specified by 'data' is moved to register 'r'.

Operation: (Rr) ← data r = 0–7

Examples:
```
MIR4:  MOV R4,#HEXTEN      ;THE VALUE OF THE SYMBOL
                           ;'HEXTEN' IS MOVED INTO REG 4
MIR 5: MOV R5,#PI*(R*R)    ;THE VALUE OF THE EXPRESSION
                           ;'PI*(R*R)' IS MOVED INTO REG 5
MIR 6: MOV R6, #0ADH       ;'AD' HEX IS MOVED INTO REG 6
```

MOV @ R_i,A Move Accumulator Contents to Data Memory

Encoding: | 1 0 1 0 | 0 0 0 i | A0H–A1H

Description: The contents of the accumulator are moved to the resident data memory location whose address is specified by bits 0–5** of register 'i'. Register 'i' contents are unaffected.

Operation: ((Ri)) ← (A) i = 0–1

Example: Assume R0 contains 00000111.
```
MDMA: MOV @R0,A            ;MOVE CONTENTS OF ACC TO
                           ;LOCATION 7 (REG 7)
```

MOV @ R_i,#data Move Immediate Data to Data memory

Encoding: | 1 0 1 1 | 0 0 0 i | | d_7 d_6 d_5 d_4 | d_3 d_2 d_1 d_0 | B0H–B1H

Description: This is a 2-cycle instruction. The 8-bit value specified by 'data' is moved to the resident data memory location addressed by register 'i', bits 0–5**.

Operation: ((Ri)) ← data i = 0–1

Examples: Move the hexadecimal value AC3F to locations 62–63.
```
MIDM:  MOV R0,#62          ;MOVE '62' DEC TO ADDR REG 0
       MOV @R0,#0ACH       ;MOVE 'AC' HEX TO LOCATION 62
       INC R0              ;INCREMENT REG 0 to '63'
       MOV @R0,#3FH        ;MOVE '3F' HEX TO LOCATION 63
```

** 0–5 in 8048AH/8748H
0–6 in 8049AH/8749H
0–7 in 8050AH

MOV T,A Move Accumulator Contents to Timer/Counter

Encoding: | 0 1 1 0 | 0 0 1 0 | 62H

Description: The contents of the accumulator are moved to the timer/event-counter register.

Operation: (T) ← (A)

Example: Initialize and start event counter.

```
INITEC: CLR A              ;CLEAR ACC TO ZEROS
        MOV T,A            ;MOVE ZEROS TO EVENT COUNTER
        START CNT          ;START COUNTER
```

MOVD A,Pp Move Port 4-7 Data to Accumulator

Encoding: | 0 0 0 0 | 1 1 p p | 0CH-0FH

Description: This is a 2-cycle instruction. Data on 8243 port 'p' is moved (read) to accumulator bits 0-3. Accumulator bits 4-7 are zeroed.

Operation: (0-3) ← (Pp) p = 4-7
(4-7) ← 0

Note: Bits 0-7 of the opcode are used to represent ports 4-7. If you are coding in binary rather than assembly language, the mapping is as follows:

Bits 1 0	Port
0 0	4
0 1	5
1 0	6
1 1	7

Example:

```
INPPT5: MOVD A,P5          ;MOVE PORT 5 DATA TO ACC
                           ;BITS 0-3, ZERO ACC BITS 4-7
```

MOVD Pp,A Move Accumulator Data to Port 4-7

Encoding: | 0 0 1 1 | 1 1 p p | 3CH-3FH

Description: This is a 2-cycle instruction. Data in accumulator bits 0-3 is moved (written) to 8243 port 'p'. Accumulator bits 4-7 are unaffected. (See NOTE above regarding port mapping.)

Operation: (Pp) ← (A_{0-3}) P = 4-7

Example: Move data in accumulator to ports 4 and 5.

```
OUTP45: MOVD P4,A          ;MOVE ACC BITS 0-3 TO PORT 4
        SWAP A             ;EXCHANGE ACC BITS 0-3 and 4-7
        MOVD P5,A          ;MOVE ACC BITS 0-3 TO PORT 5
```

MOVP A,@A Move Current Page Data to Accumulator

Encoding: | 1 0 1 0 | 0 0 1 1 | A3H

Description: The contents of the program memory location addressed by the accumulator are moved to the accumulator. Only bits 0–7 of the program counter are affected, limiting the program memory reference to the current page. The program counter is restored *following* this operation.

Operation: $(PC_{0-7}) \leftarrow (A)$
$(A) \leftarrow ((PC))$
Note: This is a 1-byte, 2-cycle instruction. If it appears in location 255 of a program memory page, @A addresses a location in the *following* page.

Example:
```
MOV128: MOV A,#128      ;MOVE '128' DEC TO ACC
        MOVP A,@A       ;CONTENTS OF 129th LOCATION IN
                        ;CURRENT PAGE ARE MOVED TO ACC
```

MOVP3 A,@A Move Page 3 Data to Accumulator

Encoding: | 1 1 1 0 | 0 0 1 1 | E3H

Description: This is a 2-cycle instruction. The contents of the program memory location (within page 3) addressed by the accumulator are moved to the accumulator. The program counter is restored following this operation.

Operation: $(PC_{0-7}) \leftarrow (A)$
$(PC_{8-11}) \leftarrow 0011$
$(A) \leftarrow ((PC))$

Example: Look up ASCII equivalent of hexadecimal code in table contained at the beginning of page 3. Note that ASCII characters are designated by a 7-bit code; the eighth bit is always reset.

```
TABSCH: MOV A,#0B8H     ;MOVE 'B8' HEX TO ACC (10111000)
        ANL A,#7FH      ;LOGICAL AND ACC TO MASK BIT
                        ;7 (00111000)
        MOVP3 A,@A      ;MOVE CONTENTS OF LOCATION '38'
                        ;HEX IN PAGE 3 TO ACC (ASCII '8')
```

Access contents of location in page 3 labelled TAB1.
Assume current program location is not in page 3.

```
TABSCH: MOV A,#LOW TAB 1 ;ISOLATE BITS 0-7 OF LABEL
                         ;ADDRESS VALUE
        MOVP3 A,@A       ;MOVE CONTENTS OF PAGE 3
                         ;LOCATION LABELED 'TAB1' TO ACC
```

MOVX A,@R_i Move External-Data-Memory Contents to Accumulator

Encoding: | 1 0 0 0 | 0 0 0 i | 80H-81H

Description: This is a 2-cycle instruction. The contents of the external data memory location addressed by register 'i' are moved to the accumulator. Register 'i' contents are unaffected. A read pulse is generated.

Operation: (A) ← ((Ri)) i = 0-1

Example: Assume R1 contains 01110110.

```
MAXDM:  MOVX A,@R1      ;MOVE CONTENTS OF LOCATION
                        ;118 TO ACC
```

MOVX @R_i,A Move Accumulator Contents to External Data Memory

Encoding: | 1 0 0 1 | 0 0 0 i | 90H-91H

Description: This is a 2-cycle instruction. The contents of the accumulator are moved to the external data memory location addressed by register 'i'. Register 'i' contents are unaffected. A write pulse is generated.

Operation: ((Ri)) ← A i = 0-1

Example: Assume R0 contains 11000111.

```
MXDMA:  MOVX @R0,A      ;MOVE CONTENTS OF ACC TO
                        ;LOCATION 199 IN EXPANDED
                        ;DATA MEMORY
```

NOP The NOP Instruction

Encoding: | 0 0 0 0 | 0 0 0 0 | 00H

Description: No operation is performed. Execution continues with the following instruction.

ORL A,R_r Logical OR Accumulator With Register Mask

Encoding: | 0 1 0 0 | 1 r r r | 48H-4FH

Description: Data in the accumulator is logically ORed with the mask contained in working register 'r'.

Operation: (A) ← (A) OR (Rr) r = 0-7

Example:

```
ORREG:  ORL A,R4        ;'OR' ACC CONTENTS WITH
                        ;MASK IN REG 4
```

ORL A,@R_i Logical OR Accumulator With Memory Mask

Encoding: | 0 1 0 0 | 0 0 0 i | 40H-41H

Description: Data in the accumulator is logically ORed with the mask contained in the resident data memory location referenced by register "i", bits 0–5**.

Operation: (A) ← (A) OR ((Ri)) i = 0-1

Example:
```
ORDM:  MOV R0,#3FH      ;MOVE '3F' HEX TO REG 0
       ORL A,@R0        ;'OR' AC CONTENTS WITH MASK
                        ;IN LOCATION 63
```

ORL A,#data Logical OR Accumulator With Immediate Mask

Encoding: | 0 1 0 0 | 0 0 1 1 | d_7 d_6 d_5 d_4 | d_3 d_2 d_1 d_0 | 43H

Description: This is a 2-cycle instruction. Data in the accumulator is logically ORed with an immediately-specified mask.

Operation: (A) ← (A) OR data

Example:
```
ORID:  ORL A,#'X'       ;'OR' ACC CONTENTS WITH MASK
                        ;01011000 (ASCII VALUE OF 'X')
```

ORL BUS,#data* Logical OR BUS With Immediate Mask

Encoding: | 1 0 0 0 | 1 0 0 0 | d_7 d_6 d_5 d_4 | d_3 d_2 d_1 d_0 | 88H

Description: This is a 2-cycle instruction. Data on the BUS port is logically ORed with an immediately-specified mask. This instruction assumes prior specification on an 'OUTL BUS,A' instruction.

Operation: (BUS) ← (BUS) OR data

Example:
```
ORBUS: ORL BUS,#HEXMSK  ;'OR' BUS CONTENTS WITH MASK
                        ;EQUAL VALUE OF SYMBOL 'HEXMSK'
```

ORL Pp, #data Logical OR Port 1 or 2 With Immediate Mask

Encoding: | 1 0 0 0 | 1 0 p p | d_7 d_6 d_5 d_4 | d_3 d_2 d_1 d_0 | 89H-8AH

Description: This is a 2-cycle instruction. Data on port 'p' is logically ORed with an immediately-specified mask.

Operation: (Pp) ← (Pp) OR data p = 1-2

Example:
```
ORP1:  ORL P1, #0FFH    ;'OR' PORT 1 CONTENTS WITH MASK
                        ;'FF' HEX (SET PORT 1 TO ALL ONES)
```

* For use with internal program memory ONLY.

** 0–5 in 8048AH/8748H
0–6 in 8049AH/8749H
0–7 in 8050AH

ORLD Pp,A Logical OR Port 4-7 With Accumulator Mask

Encoding: | 1 0 0 0 | 1 1 p p | 8CH-8FH

Description: This is a 2-cycle instruction. Data on port 'p' is logically ORed with the digit mask contained in accumulator bits 0-3.

Operation: (Pp) ← (Pp) OR (A_{0-3}) p = 4-7

Example:
```
ORP7:  ORLD P7,A      ;'OR' PORT 7 CONTENTS WITH ACC
                      ;BITS 0-3
```

OUTL BUS,A* Output Accumulator Data to BUS

Encoding: | 0 0 0 0 | 0 0 1 0 | 02H

Description: This is a 2-cycle instruction. Data residing in the accumulator is transferred (written) to the BUS port and latched. The latched data remains valid until altered by another OUTL instruction. Any other instruction requiring use of the BUS port (except INS) destroys the contents of the BUS latch. This includes expanded memory operations (such as the MOVX instruction). Logical operations on BUS data (AND, OR) assume the OUTL BUS,A instruction has been issued previously.

Operation: (BUS) ← (A)

Example:
```
OUTLBP:  OUTL BUS, A      ;OUTPUT ACC CONTENTS TO BUS
```

OUTL Pp,A Output Accumulator Data to Port 1 or 2

Encoding: | 0 0 1 1 | 1 0 p p | 39H-3AH

Description: This is a 2-cycle instruction. Data residing in the accumulator is transferred (written) to port 'p' and latched.

Operation: (Pp) ← (A) p = 1-2

Example:
```
OUTLP:  MOV A,R7      ;MOVE REG 7 CONTENTS TO ACC
        OUTL P2,A     ;OUTPUT ACC CONTENTS TO PORT 2
        MOV A, R6     ;MOV REG 6 CONTENTS TO ACC
        OUTL P1,A     ;OUTPUT ACC CONTENTS TO PORT 1
```

* For use with internal program memory ONLY.

RET Return Without PSW Restore

Encoding: | 1 0 0 0 | 0 0 1 1 | 83H

Description: This is a 2-cycle instruction. The stack pointer (PSW bits 0–2) is decremented. The program counter is then restored from the stack. PSW bits 4–7 are not restored.

Operation: (SP) ← (SP)–1
(PC) ← ((SP))

RETR Return with PSW Restore

Encoding: | 1 0 0 1 | 0 0 1 1 | 93H

Description: This is a 2-cycle instruction. The stack pointer is decremented. The program counter and bits 4–7 of the PSW are then restored from the stack. Note that RETR should be used to return from an interrupt, but should not be used within the interrupt service routine as it signals the end of an interrupt routine by resetting the Interrupt in Progress flip-flop.

Operation: (SP) ← (SP)–1
(PC) ← ((SP))
(PSW 4–7) ← ((SP))

RL A Rotate Left without Carry

Encoding: | 1 1 1 0 | 0 1 1 1 | E7H

Description: The contents of the accumulator are rotated left one bit. Bit 7 is rotated into the bit 0 position.

Operation: (An + 1) ← (An)
(A0) ← (A7) n = 0–6

Example: Assume accumulator contains 10110001.

```
RLNC: RL A          ;NEW ACC CONTENTS ARE 01100011
```

RLC A Rotate Left through Carry

Encoding: | 1 1 1 1 | 0 1 1 1 | F7H

Description: The contents of the accumulator are rotated left one bit. Bit 7 replaces the carry bit; the carry bit is rotated into the bit 0 position.

Operation: (An + 1) ← (An)
n = 0–6
(A0) ← (C)
(C) ← (A7)

Example: Assume accumulator contains a 'signed' number; isolate sign without changing value

```
RLTC: CLR C         ;CLEAR CARRY TO ZERO
      RLC A         ;ROTATE ACC LEFT, SIGN
                    ;BIT(7) IS PLACED IN CARRY
      RR A          ;ROTATE ACC RIGHT — VALUE
                    ;(BITS 0-6) IS RESTORED,
                    ;CARRY UNCHANGED, BIT 7
                    ;IS ZERO
```

RR A Rotate Right without Carry

Encoding: | 0 1 1 1 | 0 1 1 1 | 77H

Description: The contents of the accumulator are rotated right one bit. Bit 0 is rotated into the bit 7 position.

Operation: (An) ← (An + 1) n = 0–6
(A7) ← (A0)

Example: Assume accumulator contains 10110001.

```
RRNC: RR A          ;NEW ACC CONTENTS ARE 11011000
```

RRC A Rotate Right through Carry

Encoding: | 0 1 1 0 | 0 1 1 1 | 67H

Description: The contents of the accumulator are rotated right one bit. Bit 0 replaces the carry bit; the carry bit is rotated into the bit 7 position.

Operation: (An) ← (An + 1) n = 0–6
(A7) ← (C)
(C) ← (A_0)

Example: Assume carry is not set and accumulator contains 10110001.

```
RRTC:  RRC A          ;CARRY IS SET AND ACC
                      ;CONTAINS 01011000
```

SEL MB0 Select Memory Bank 0

Encoding: | 1 1 1 0 | 0 1 0 1 | E5H

Description: PC bit 11 is set to zero on next JMP or CALL instruction. All references to program memory addresses fall within the range 0–2047.

Operation: (DBF) ← 0

Example: Assume program counter contains 834 Hex.

```
SEL MB0               ;SELECT MEMORY BANK 0
JMP $+20              ;JUMP TO LOCATION 58 HEX
```

SEL MB1 Select Memory Bank 1

Encoding: | 1 1 1 1 | 0 1 0 1 | F5H

Description: PC bit 11 is set to one on next JMP or CALL instruction. All references to program memory addresses fall within the range 2048–4095.

Operation: (DBF) ← 1

SEL RB0 Select Register Bank 0

Encoding: | 1 1 0 0 | 0 1 0 1 | C5H

Description: PSW bit 4 is set to zero. References to working registers 0-7 address data memory locations 0-7. This is the recommended setting for normal program execution.

Operation: (BS) ← 0

SEL RB1 Select Register Bank 1

Encoding: | 1 1 0 1 | 0 1 0 1 | D5H

Description: PSW bit 4 is set to one. References to working registers 0-7 address data memory locations 24-31. This is the recommended setting for interrupt service routines, since locations 0-7 are left intact. The setting of PSW bit 4 in effect at the time of an interrupt is restored by the RETR instruction when the interrupt service routine is completed.

Operation: (BS) ← 1

Example: Assume an external interrupt has occurred, control has passed to program memory location 3, and PSW bit 4 was zero before the interrupt.

Operation:

```
LOC3: JNI INIT          ;JUMP TO ROUTINE 'INIT' IF
                        ;INTERRUPT INPUT IS ZERO
INIT: MOV R7,A          ;MOVE ACC CONTENTS TO
                        ;LOCATION 7
      SEL RB1           ;SELECT REG BANK 1
      MOV R7,#0FAH      ;MOVE 'FA' HEX TO LOCATION 31
      .
      .
      .
      SEL RB0           ;SELECT REG BANK 0
      MOV A,R7          ;RESTORE ACC FROM LOCATION 7
      RETR              ;RETURN — RESTORE PC AND PSW
```

STOP TCNT Stop Timer/Event-Counter

Encoding: | 0 1 1 0 | 0 1 0 1 | 65H

Description: This instruction is used to stop both time accumulation and event counting.

Example: Disable interrupt, but jump to interrupt routine after eight overflows and stop timer. Count overflows in register 7.

```
START:  DIS TCNTI          ;DISABLE TIMER INTERRUPT
        CLR A              ;CLEAR ACC TO ZEROS
        MOV T,A            ;MOVE ZEROS TO TIMER
        MOV R7,A           ;MOVE ZEROS TO REG 7
        STRT T             ;START TIMER
MAIN:   JTF COUNT          ;JUMP TO ROUTINE 'COUNT'
                           ;IF TF = 1 AND CLEAR TIMER FLAG
        JMP MAIN           ;CLOSE LOOP
COUNT:  INC R7             ;INCREMENT REG 7
        MOV A,R7           ;MOVE REG 7 CONTENTS TO ACC
        JB3 INT            ;JUMP TO ROUTINE 'INT' IF ACC
                           ;BIT 3 IS SET (REG 7 = 8)
        JMP MAIN           ;OTHERWISE RETURN TO ROUTINE
                           ;MAIN
        .
        .
        .
INT:    STOP TCNT          ;STOP TIMER
        JMP 7H             ;JUMP TO LOCATION 7 (TIMER)
                           ;INTERRUPT ROUTINE
```

STRT CNT Start Event Conter

Encoding:

0 1 0 0	0 1 0 1

45H

Description: The test 1 (T1) pin is enabled as the event-counter input and the counter is started. The event-counter register is incremented with each high-to-low transition on the T1 pin.

Example: Initialize and start event counter. Assume overflow is desired with first T1 input.

```
STARTC: EN TCNTI           ;ENABLE COUNTER INTERRUPT
        MOV A,#0FFH        ;MOVE 'FF'HEX (ONES) TO ACC
        MOV T,A            ;MOVES ONES TO COUNTER
        STRT CNT           ;ENABLE T1 AS COUNTER
                           ;INPUT AND START
```

STRT T Start Timer

Encoding: | 0 1 0 1 | 0 1 0 1 | 55H

Description: Timer accumulation is initiated in the timer register. The register is incremented every 32 instruction cycles. The prescaler which counts the 32 cycles is cleared but the timer register is not.

Example: Initialize and start timer.

```
STARTT: CLR A          ;CLEAR ACC TO ZEROS
        MOV T,A        ;MOVE ZEROS TO TIMER
        EN TCNTI       ;ENABLE TIMER INTERRUPT
        STRT T         ;START TIMER
```

SWAP A Swap Nibbles within Accumulator

Encoding: | 0 1 0 0 | 0 1 1 1 | 47H

Description: Bits 0-3 of the accumulator are swapped with bits 4-7 of the accumulator.

Operation: $(A_{4-7}) \rightleftarrows (A_{0-3})$

Example: Pack bits 0-3 of locations 50-51 into location 50.

```
PCKDIG: MOV R0, #50     ;MOVE '50' DEC TO REG 0
        MOV R1, #51     ;MOVE '51' DEC TO REG 1
        XCHD A,@R0      ;EXCHANGE BITS 0-3 OF ACC
                        ;AND LOCATION 50
        SWAP A          ;SWAP BITS 0-3 AND 4-7 OF ACC
        XCHD A,@R1      ;EXCHANGE BITS 0-3 OF ACC AND
                        ;LOCATION 51
        MOV @R0,A       ;MOVE CONTENTS OF ACC TO
                        ;LOCATION 50
```

XCH A,R_r Exchange Accumulator-Register Contents

Encoding: | 0 0 1 0 | 1 r r r | 28H-2FH

Description: The contents of the accumulator and the contents of working register 'r' are exchanged.

Operation: $(A) \rightleftarrows (Rr)$ r = 0-7

Example: Move PSW contents to Reg 7 without losing accumulator contents.

```
XCHAR7: XCH A,R7        ;EXCHANGE CONTENTS OF REG 7
                        ;AND ACC
        MOV A, PSW      ;MOVE PSW CONTENTS TO ACC
        XCH A,R7        ;EXCHANGE CONTENTS OF REG 7
                        'AND ACC AGAIN
```

XCH A,@R_i Exchange Accumulator and Data Memory Contents

Encoding: | 0 0 1 0 | 0 0 0 i | 20H-21H

Description: The contents of the accumulator and the contents of the resident data memory location addressed by bits 0-5** of register 'i' are exchanged. Register 'i' contents are unaffected.

Operation: (A) ⇆ ((Ri)) i = 0-1

Example: Decrement contents of location 52.

```
DEC52: MOV R0,#52      ;MOVE '52' DEC TO ADDRESS REG 0
       XCH A,@R0       ;EXCHANGE CONTENTS OF ACC
                       ;AND LOCATION 52
       DEC A           ;DECREMENT ACC CONTENTS
       XCH A,@R0       ;EXCHANGE CONTENTS OF ACC
                       ;AND LOCATION 52 AGAIN
```

XCHD A,@R_i Exchange Accumulator and Data Memory 4-Bit Data

Encoding: | 0 0 1 1 | 0 0 0 i | 30H-31H

Description: This instruction exchanges bits 0-3 of the accumulator with bits 0-3 of the data memory location addressed by bits 0-5** of register 'i'. Bits 4-7 of the accumulator, bits 4-7 of the data memory location, and the contents of register 'i' are unaffected.

Operation: (A_{0-3}) ⇆ ((Ri0-3)) i = 0-1

Example: Assume program counter contents have been stacked in locations 22-23.

```
XCHNIB: MOV R0,#23     ;MOVE '23' DEC TO REG 0
        CLR A          ;CLEAR ACC TO ZEROS
        XCHD A,@R0     ;EXCHANGE BITS 0-3 OF ACC AND
                       ;LOCATION 23 (BTS 8-11 OF PC ARE
                       ;ZEROED, ADDRESS REFERS
                       ;TO PAGE 0)
```

XRL A,R_r Logical XOR Accumulator With Register Mask

Encoding: | 1 1 0 1 | 1 r r r | D8H-DFH

Description: Data in the accumulator is EXCLUSIVE ORed with the mask contained in working register 'r'.

Operation: (A) ← (A) XOR (Rr) r = 0-7

Example:

```
XORREG: XRL A,R5       ;'XOR' ACC CONTENTS WITH
                       ;MASK IN REG 5
```

** 0-5 in 8048AH/8748H
0-6 in 8049AH/8749H
0-7 in 8050AH

XRL A,@R_i Logical XOR Accumulator With Memory Mask

Encoding: | 1 1 0 1 | 0 0 0 i | D0H-D1H

Description: Data in the accumulator is EXCLUSIVE ORed with the mask contained in the data memory location addressed by register 'i', bits 0–5.**

Operation: (A) ← (A) XOR ((Ri)) i = 0–1

Example:
```
XORDM:  MOV R1,#20H     ;MOVE '20' HEX TO REG 1
        XRL A,@R1       ;'XOR' ACC CONTENTS WITH MASK
                        ;IN LOCATION 32
```

XRL A,#data Logical XOR Accumulator With Immediate Mask

Encoding: | 1 1 0 1 | 0 0 1 1 | | d_7 d_6 d_5 d_4 | d_3 d_2 d_1 d_0 | D3H

Description: This is a 2-cycle instruction. Data in the accumulator is EXCLUSIVE ORed with an immediately-specified mask.

Operation: (A) ← (A) XOR data

Example:
```
XORID:  XOR A,#HEXTEN   ;XOR CONTENTS OF ACC WITH MASK
                        ;EQUAL VALUE OF SYMBOL 'HEXTEN'
```

** 0–5 in 8048AH/8748H
0–6 in 8049AH/8749H
0–7 in 8050AH

and Index

MCS®-45 Data Sheet and Index

4

8243
MCS®-48 INPUT/OUTPUT EXPANDER

■ 0°C TO 70°C Operation

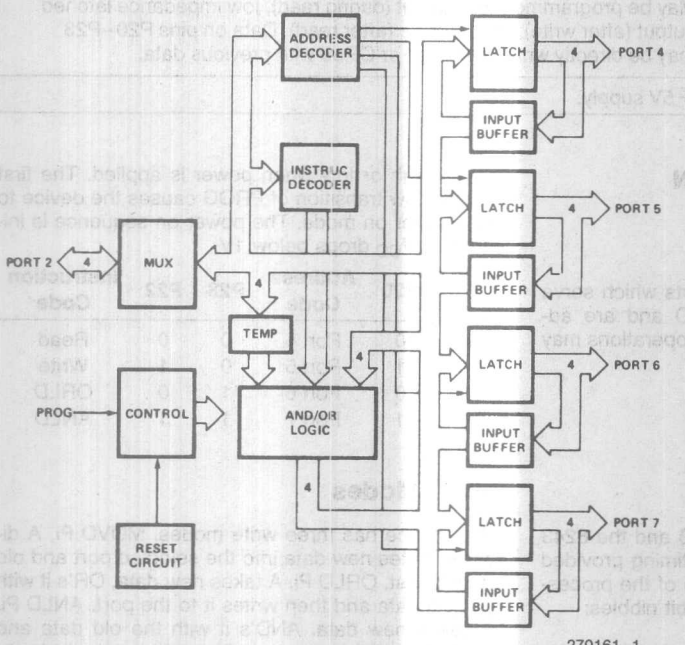


Figure 1. 8243 Block Diagram

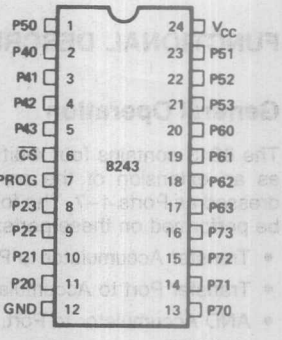


Figure 2. 8243 Pin Configuration

October 1986
Order Number: 270161-001

Table 1. Pin Description

Symbol	Pin No.	Function
PROG	7	Clock Input. A high to low transition on PROG signifies that address and control are available on P20–P23, and a low to high transition signifies that data is available on P20–P23.
$\overline{CS}$	6	Chip Select Input. A high on CS inhibits any change of output or internal status.
P20–P23	11–8	Four (4) bit bi-directional port contains the address and control bits on a high to low transition of PROG. During a low to high transition, P2 contains the data for a selected output port if a write operation, or the data from a selected port before the low to high transition if a read operation.
GND	12	0V supply.
P40–P43	2–5	Four (4) bit bi-directional I/O ports.
P50–P53 P60–P63 P70–P73	1, 23–21 20–17 13–16	May be programmed to be input (during read), low impedance latched output (after write), or a tri-state (after read). Data on pins P20–P23 may be directly written, ANDed or ORed with previous data.
V_{CC}	24	+5V supply.

FUNCTIONAL DESCRIPTION

General Operation

The 8243 contains four 4-bit I/O ports which serve as an extension of the on-chip I/O and are addressed as Ports 4–7. The following operations may be performed on these ports:

- Transfer Accumulator to Port.
- Transfer Port to Accumulator.
- AND Accumulator to Port.
- OR Accumulator to Port.

All communication between the 8048 and the 8243 occurs over Port 2 (P20–P23) with timing provided by an output pulse on the PROG pin of the processor. Each transfer consists of two 4-bit nibbles:

The first containing the "op code" and port address and the second containing the actual 4-bits of data. A high to low transition of the PROG line indicates that address is present while a low to high transition indicates the presence of data. Additional 8243's may be added to the 4-bit bus and chip selected using additional output lines from the 8048/8748/8035.

Power On Initialization

Initial application of power to the device forces input/output Ports 4, 5, 6, and 7 to the tri-state and Port 2 to the input mode. The PROG pin may be either high or low when power is applied. The first high to low transition of PROG causes the device to exit power on mode. The power on sequence is initiated if V_{CC} drops below 1V.

P21	P20	Address Code	P23	P22	Instruction Code
0	0	Port 4	0	0	Read
0	1	Port 5	0	1	Write
1	0	Port 6	1	0	ORLD
1	1	Port 7	1	1	ANLD

Write Modes

The device has three write modes. MOVD Pi, A directly writes new data into the selected port and old data is lost. ORLD Pi, A takes new data, OR's it with the old data and then writes it to the port. ANLD Pi, A takes new data, AND's it with the old data and then writes it to the port. Operation code and port address are latched from the input Port 2 on the high to low transition of the PROG pin. On the low to high transition of PROG data on Port 2 is transferred to the logic block of the specified output port.

After the logic manipulation is performed, the data is latched and outputed. The old data remains latched until new valid outputs are entered.

Read Mode

The device has one read mode. The operation code and port address are latched from the input Port 2

on the high to low transition of the PROG pin. As soon as the read operation and port address are decoded, the appropriate outputs are tri-stated, and the input buffers switched on. The read operation is terminated by a low to high transition of the PROG pin. The port (4, 5, 6 or 7) that was selected is switched to the tri-stated mode while Port 2 is returned to the input mode.

Normally, a port will be in an output (write mode) or input (read mode). If modes are changed during operation, the first read following a write should be ignored; all following reads are valid. This is to allow the external driver on the port to settle after the first read instruction removes the low impedance drive from the 8243 output. A read of any port will leave that port in a high impedance state.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C

Storage Temperature .....−65°C to +150°C

Voltage on Any Pin with Respect to Ground.....−0.5V to +7V

Power Dissipation .....1 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ± 10%

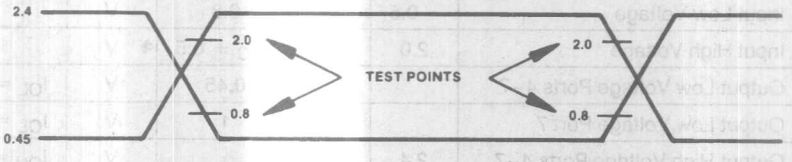
Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC} + 0.5	V	
V_{OL1}	Output Low Voltage Ports 4–7			0.45	V	I_{OL} = 4.5 mA*
V_{OL2}	Output Low Voltage Port 7			1	V	I_{OL} = 20 mA
V_{OH1}	Output High Voltage Ports 4–7	2.4			V	I_{OH} = 240 μA
I_{IL1}	Input Leakage Ports 4–7	−10		20	μA	V_{in} = V_{CC} to 0V
I_{IL2}	Input Leakage Port 2, CS, PROG	−10		10	μA	V_{in} = V_{CC} to 0V
V_{OL3}	Output Low Voltage Port 2			0.45	V	I_{OL} = 0.6 mA
I_{CC}	V_{CC} Supply Current		10	20	mA	(Note 1)
V_{OH2}	Output Voltage Port 2	2.4				I_{OH} = 100 μA
I_{OL}	Sum of All I_{OL} From 16 Outputs			72	mA	4.5 mA Each Pin

*Refer to Figure 3 for additional sink current capability.

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ± 10%

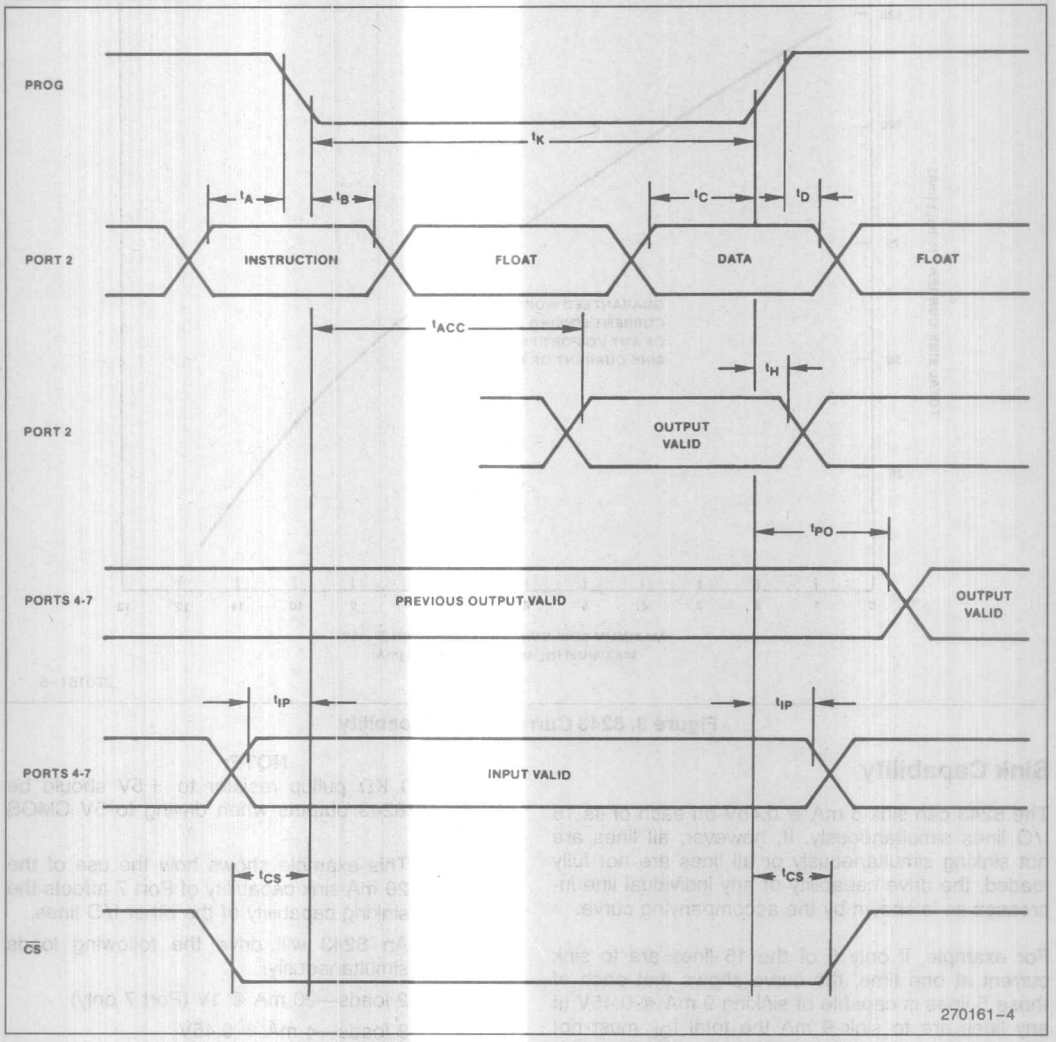
Symbol	Parameter	Min	Max	Units	Test Conditions
t_A	Code Valid before PROG	50		ns	80 pF Load
t_B	Code Valid after PROG	60		ns	20 pF Load
t_C	Data Valid before PROG	200		ns	80 pF Load
t_D	Data Valid after PROG	20		ns	20 pF Load
t_H	Floating after PROG	0	150	ns	20 pF Load
t_K	PROG Negative Pulse Width	700		ns	
t_{CS}	CS Valid before/after PROG	50		ns	
t_{PO}	Ports 4–7 Valid after PROG		700	ns	100 pF Load
t_{LP1}	Ports 4–7 Valid before/after PROG	100		ns	
t_{ACC}	Port 2 Valid after PROG		650	ns	80 pF Load

NOTE:
1. I_{CC} (−40°C to 85°C EXPRESS options) 15 mA typical/25 mA maximum.



270161-3

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Output timing measurements are made at 2.0V for Logic "1" and 0.8V for a Logic "0".


PROG
t_K
t_A
t_B
t_C
t_D
PORT 2
INSTRUCTION
FLOAT
DATA
FLOAT
t_{ACC}
t_H
PORT 2
OUTPUT VALID
t_{PO}
PORTS 4-7
PREVIOUS OUTPUT VALID
OUTPUT VALID
t_{IP}
t_{IP}
PORTS 4-7
INPUT VALID
t_{CS}
t_{CS}
$\overline{CS}$
270161-4

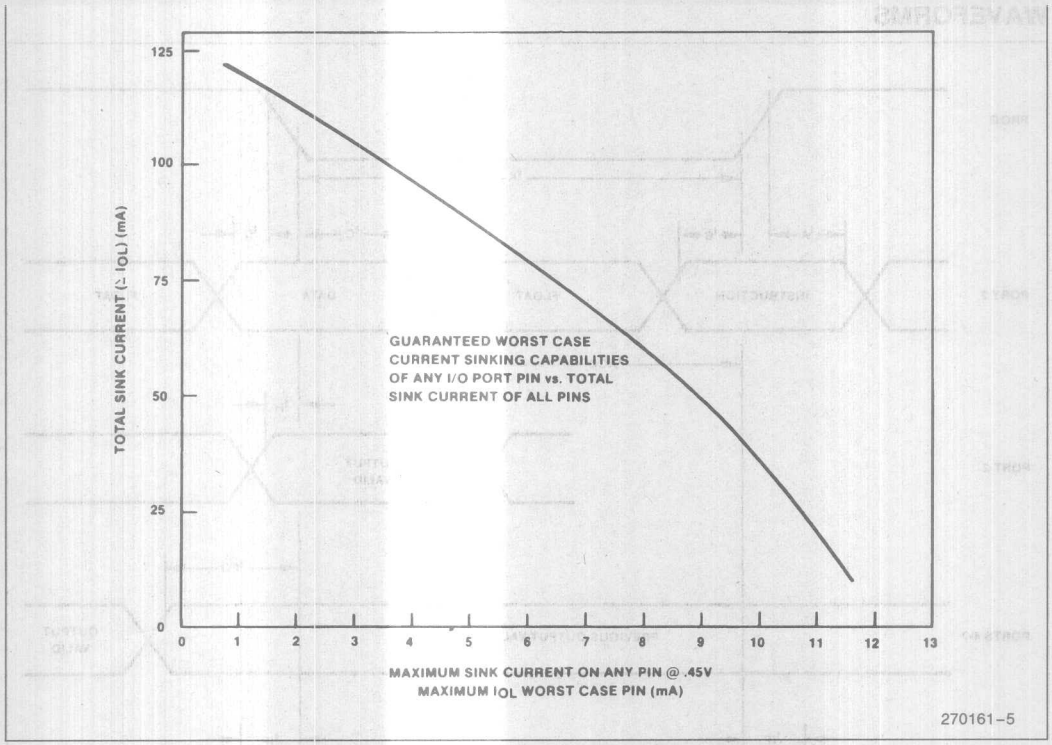


Figure 3. 8243 Current Sink Capability

Sink Capability

The 8243 can sink 5 mA @ 0.45V on each of its 16 I/O lines simultaneously. If, however, all lines are not sinking simultaneously or all lines are not fully loaded, the drive capability of any individual line increases as is shown by the accompanying curve.

For example, if only 5 of the 16 lines are to sink current at one time, the curve shows that each of those 5 lines is capable of sinking 9 mA @ 0.45V (if any lines are to sink 9 mA the total I_{OL} must not exceed 45 mA or five 9 mA loads).

Example: How many pins can drive 5 TTL loads (1.6 mA) assuming remaining pins are unloaded?

$I_{OL} = 5 \times 1.6\ mA = 8\ mA$

$\epsilon I_{OL} = 60$ mA from curve

pins $= 60\ mA \div 8\ mA/pin = 7.5 = 7$

In this case, 7 lines can sink 8 mA for a total of 56 mA. This leaves 4 mA sink current capability which can be divided in any way among the remaining 8 I/O lines of the 8243.

NOTE:

A10 to 50 KΩ pullup resistor to +5V should be added to 8243 outputs when driving to 5V CMOS directly.

Example: This example shows how the use of the 20 mA sink capability of Port 7 affects the sinking capability of the other I/O lines.

An 8243 will drive the following loads simultaneously.

2 loads—20 mA @ 1V (Port 7 only)

8 loads—4 mA @ 0.45V

6 loads—3.2 mA @ 0.45V

Is this within the specified limits?

$\epsilon I_{OL} = (2 \times 20) + (8 \times 4) + (6 \times 3.2) = 91.2$ mA.

From the curve: for $I_{OL} = 4$ mA, $\epsilon I_{OL} \cong 93$ mA. Since 91.2 mA < 93 mA the loads are within specified limits.

Although the 20 mA @ 1V loads are used in calculating ϵI_{OL}, it is the largest current required @ 0.45V which determines the maximum allowable ϵI_{OL}.

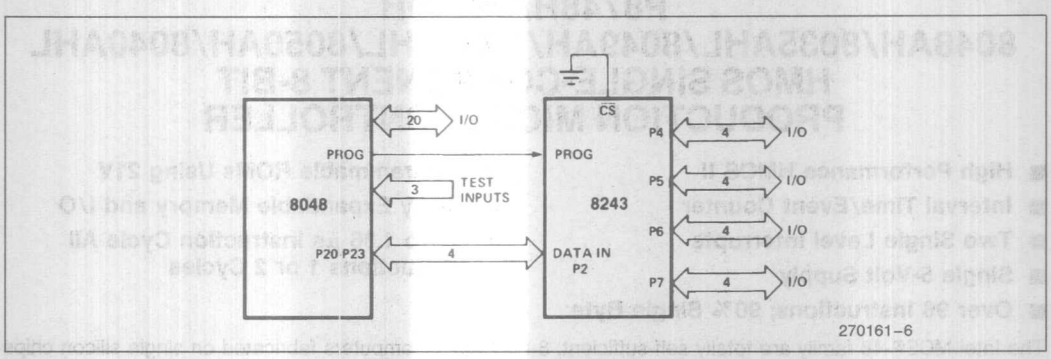


Figure 4. Expander Interface

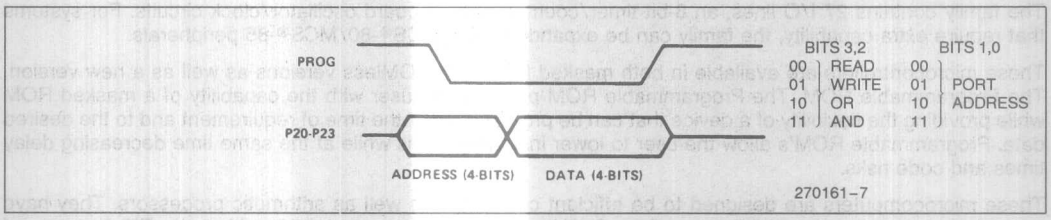


Figure 5. Output Expander Timing

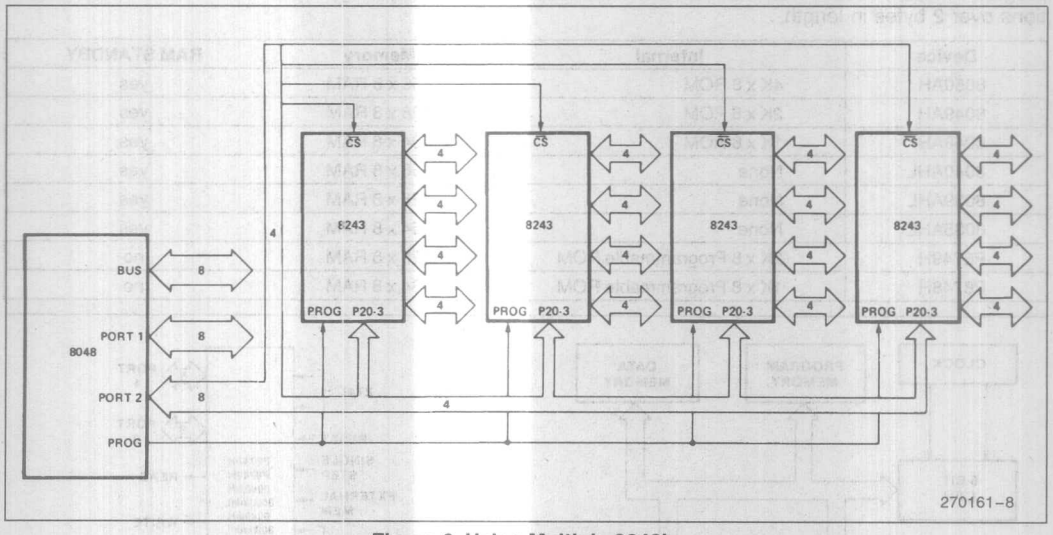


Figure 6. Using Multiple 8243's

P8748H/P8749H
8048AH/8035AHL/8049AH/8039AHL/8050AH/8040AHL
HMOS SINGLE-COMPONENT 8-BIT PRODUCTION MICROCONTROLLER

- **High Performance HMOS II**
- **Interval Time/Event Counter**
- **Two Single Level Interrupts**
- **Single 5-Volt Supply**
- **Over 96 Instructions; 90% Single Byte**
- **Programmable ROMs Using 21V**
- **Easily Expandable Memory and I/O**
- **Up to 1.36 μs Instruction Cycle All Instructions 1 or 2 Cycles**

The Intel MCS®-48 family are totally self-sufficient, 8-bit parallel computers fabricated on single silicon chips using Intel's advanced N-channel silicon gate HMOS process.

The family contains 27 I/O lines, an 8-bit timer/counter, and on-board oscillator/clock circuits. For systems that require extra capability, the family can be expanded using MCS®-80/MCS®-85 peripherals.

These microcontrollers are available in both masked ROM and ROMless versions as well as a new version, The Programmable ROM. The Programmable ROM provides the user with the capability of a masked ROM while providing the flexibility of a device that can be programmed at the time of requirement and to the desired data. Programmable ROM's allow the user to lower inventory levels while at the same time decreasing delay times and code risks.

These microcomputers are designed to be efficient controllers as well as arithmetic processors. They have extensive bit handling capability as well as facilities for both binary and BCD arithmetic. Efficient use of program memory results from an instruction set consisting of mostly single byte instructions and no instructions over 2 bytes in length.

Device	Internal	Memory	RAM STANDBY
8050AH	4K x 8 ROM	256 x 8 RAM	yes
8049AH	2K x 8 ROM	128 x 8 RAM	yes
8048AH	1K x 8 ROM	64 x 8 RAM	yes
8040AHL	None	256 x 8 RAM	yes
8039AHL	None	128 x 8 RAM	yes
8035AHL	None	64 x 8 RAM	yes
P8749H	2K x 8 Programmable ROM	128 x 8 RAM	no
P8748H	1K x 8 Programmable ROM	64 x 8 RAM	no

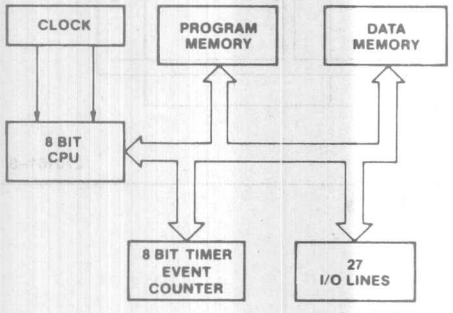


Figure 1. Block Diagram

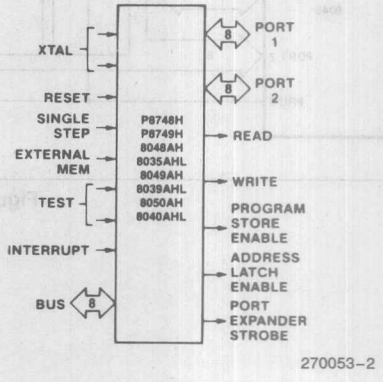


Figure 2. Logic Symbol

November 1987
Order Number: 270053-002

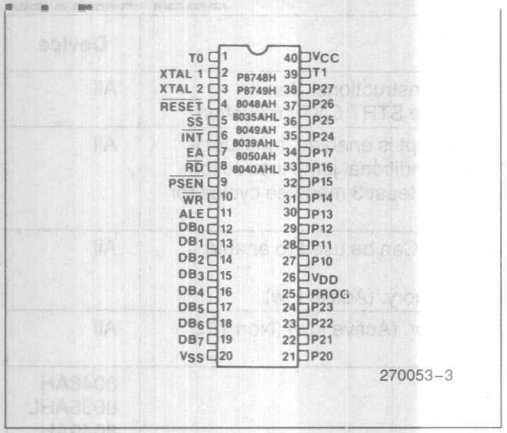


Figure 3. Pin Configuration

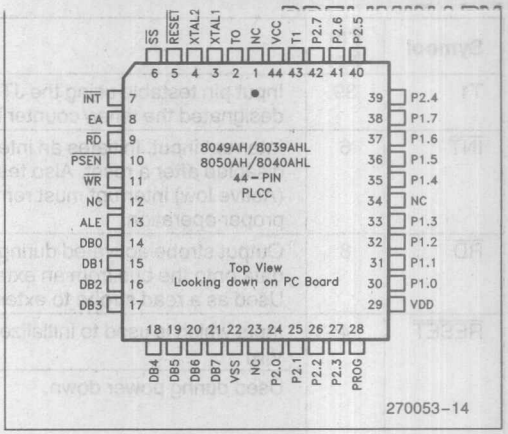


Figure 4. Pad Configuration

Table 1. Pin Description

Symbol	Pin No.	Function	Device
V_{SS}	20	Circuit GND potential.	All
V_{DD}	26	+5V during normal operation.	All
		Low power standby pin.	8048AH 8035AHL 8049AH 8039AHL 8050AH 8040AHL
		Programming power supply (+21V).	P8748H P8749H
V_{CC}	40	Main power supply; +5V during operation and programming.	All
PROG	25	Output strobe for 8243 I/O expander.	All
		Program pulse (+18V) input pin During Programming.	P8748H P8749H
P10–P17 Port 1	27–34	8-bit quasi-bidirectional port.	All
P20–P23 P24–P27 Port 2	21–24 35–38	8-bit quasi-bidirectional port. P20–P23 contain the four high order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for 8243.	All
DB0–DB7 BUS	12–19	True bidirectional port which can be written or read synchronously using the $\overline{RD}$, $\overline{WR}$ strobes. The port can also be statically latched. Contains the 8 low order program counter bits during an external program memory fetch, and receives the addressed instruction under the control of $\overline{PSEN}$. Also contains the address and data during an external RAM data store instruction, under control of ALE, $\overline{RD}$, and $\overline{WR}$.	All
T0	1	Input pin testable using the conditional transfer instruction JT0 and JNT0. T0 can be designated as a clock output using ENT0 CLK instruction.	All
		Used during programming.	P8748H P8749H

Table 1. Pin Description (Continued)

Symbol	Pin No.	Function	Device
T1	39	Input pin testable using the JT1, and JNT1 instructions. Can be designated the timer/counter input using the STRT CNT instruction.	All
$\overline{INT}$	6	Interrupt input. Initiates an interrupt if interrupt is enabled. Interrupt is disabled after a reset. Also testable with conditional jump instruction. (Active low) interrupt must remain low for at least 3 machine cycles for proper operation.	All
$\overline{RD}$	8	Output strobe activated during a BUS read. Can be used to enable data onto the bus from an external device. Used as a read strobe to external data memory. (Active low)	All
$\overline{RESET}$	4	Input which is used to initialize the processor. (Active low) (Non TTL V_{IH})	All
		Used during power down.	8048AH 8035AHL 8049AH 8039AHL 8050AH 8040AHL
		Used during programming.	P8748H P8749H
		Used during ROM verification.	8048AH P8748H 8049AH P8749H 8050AH
$\overline{WR}$	10	Output strobe during a bus write. (Active low) Used as write strobe to external data memory.	All
ALE	11	Address latch enable. This signal occurs once during each cycle and is useful as a clock output. The negative edge of ALE strobes address into external data and program memory.	All
$\overline{PSEN}$	9	Program store enable. This output occurs only during a fetch to external program memory. (Active low)	All
$\overline{SS}$	5	Single step input can be used in conjunction with ALE to "single step" the processor through each instruction.	All
		(Active low) Used in sync mode.	8048AH 8035AHL 8049AH 8039AHL 8050AH 8040AHL
EA	7	External access input which forces all program memory fetches to reference external memory. Useful for emulation and debug. (Active high)	All
		Used during (18V) programming.	P8748H P8749H
		Used during ROM verification (12V).	8048AH 8049AH 8050AH
XTAL1	2	One side of crystal input for internal oscillator. Also input for external source. (Non TTL V_{IH})	All
XTAL2	3	Other side of crystal input.	All

Table 2. Instruction Set

Accumulator

Mnemonic	Description	Bytes	Cycles
ADD A, R	Add register to A	1	1
ADD A, @R	Add data memory to A	1	1
ADD A, #data	Add immediate to A	2	2
ADDC A, R	Add register with carry	1	1
ADDC A, @R	Add data memory with carry	1	1
ADDC A, #data	Add immediate with carry	2	2
ANL A, R	And register to A	1	1
ANL A, @R	And data memory to A	1	1
ANL A, #data	And immediate to A	2	2
ORL A, R	Or register to A	1	1
ORL A, @R	Or data memory to A	1	1
ORL A, #data	Or immediate to A	2	2
XRL A, R	Exclusive or register to A	1	1
XRL A, @R	Exclusive or data memory to A	1	1
XRL A, #data	Exclusive or immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1

Input/Output

Mnemonic	Description	Bytes	Cycles
IN A, P	Input port to A	1	2
OUTL P, A	Output A to port	1	2
ANL P, #data	And immediate to port	2	2
ORL P, #data	Or immediate to port	2	2
INS A, BUS	Input BUS to A	1	2
OUTL BUS, A	Output A to BUS	1	2
ANL BUS, #data	And immediate to BUS	2	2
ORL BUS, #data	Or immediate to BUS	2	2
MOVD A, P	Input expander port to A	1	2
MOVD P, A	Output A to expander port	1	2
ANLD P, A	And A to expander port	1	2
ORLD P, A	Or A to expander port	1	2

Registers

Mnemonic	Description	Bytes	Cycles
INC R	Increment register	1	1
INC @R	Increment data memory	1	1
DEC R	Decrement register	1	1

Branch

Mnemonic	Description	Bytes	Cycles
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R, addr	Decrement register and skip	2	2
JC addr	Jump on carry = 1	2	2
JNC addr	Jump on carry = 0	2	2
JZ addr	Jump on A zero	2	2
JNZ addr	Jump on A not zero	2	2
JT0 addr	Jump on T0 = 1	2	2
JNT0 addr	Jump on T0 = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 = 1	2	2
JF1 addr	Jump on F1 = 1	2	2
JTF addr	Jump on timer flag	2	2
JNI addr	Jump on $\overline{INT}$ = 0	2	2
JBb addr	Jump on accumulator bit	2	2

Table 2. Instruction Set (Continued)

Subroutine

Mnemonic	Description	Bytes	Cycles
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2

Flags

Mnemonic	Description	Bytes	Cycles
CLR C	Clear carry	1	1
CPL C	Complement carry	1	1
CLR F0	Clear flag 0	1	1
CPL F0	Complement flag 0	1	1
CLR F1	Clear flag 1	1	1
CPL F1	Complement flag 1	1	1

Data Moves

Mnemonic	Description	Bytes	Cycles
MOV A, R	Move register to A	1	1
MOV A, @R	Move data memory to A	1	1
MOV A, #data	Move immediate to A	2	2
MOV R, A	Move A to register	1	1
MOV @R, A	Move A to data memory	1	1
MOV R, #data	Move immediate to register	2	2
MOV @R, #data	Move immediate to data memory	2	2
MOV A, PSW	Move PSW to A	1	1
MOV PSW, A	Move A to PSW	1	1
XCH A, R	Exchange A and register	1	1
XCH A, @R	Exchange A and data memory	1	1
XCHD A, @R	Exchange nibble of A and data memory	1	1
MOVX A, @R	Move external data memory to A	1	2
MOVX @R, A	Move A to external data memory	1	2
MOVP A, @A	Move to A from current page	1	2
MOVP3 A, @A	Move to A from page 3	1	2

Timer/Counter

Mnemonic	Description	Bytes	Cycles
MOV A, T	Read timer/counter	1	1
MOV T, A	Load timer/counter	1	1
STRT T	Start timer	1	1
STRT CNT	Start counter	1	1
STOP TCNT	Stop timer/counter	1	1
EN TCNTI	Enable timer/counter interrupt	1	1
DIS TCNTI	Disable timer/counter interrupt	1	1

Control

Mnemonic	Description	Bytes	Cycles
EN I	Enable external interrupt	1	1
DIS I	Disable external interrupt	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
SEL MB0	Select memory bank 0	1	1
SEL MB1	Select memory bank 1	1	1
ENT0 CLK	Enable clock output on T0	1	1

Mnemonic	Description	Bytes	Cycles
NOP	No operation	1	1

ABSOLUTE MAXIMUM RATINGS*

Case Temperature Under Bias ..... 0°C to +70°C
Storage Temperature ..... −65°C to +150°C
Voltage on any Pin with Respect to Ground ..... −0.5V to +7V
Power Dissipation 1.5W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS T_A = 0°C to +70°C; V_{CC} = V_{DD} = 5V ±10%; V_{SS} = 0V

Symbol	Parameter	Limits			Unit	Test Conditions	Device
		Min	Typ	Max			
V_{IL}	Input Low Voltage (All Except $\overline{RESET}$, X1, X2)	−0.5		0.8	V		All
V_{IL1}	Input Low Voltage ($\overline{RESET}$, X1, X2)	−5		0.6	V		All
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)	2.0		V_{CC}	V		All
V_{IH1}	Input High Voltage (X1, X2, $\overline{RESET}$)	3.8		V_{CC}	V		All
V_{OL}	Output Low Voltage (BUS)			0.45	V	I_{OL} = 2.0 mA	All
V_{OL1}	Output Low Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)			0.45	V	I_{OL} = 1.8 mA	All
V_{OL2}	Output Low Voltage (PROG)			0.45	V	I_{OL} = 1.0 mA	All
V_{OL3}	Output Low Voltage (All Other Outputs)			0.45	V	I_{OL} = 1.6 mA	All
V_{OH}	Output High Voltage (BUS)	2.4			V	I_{OH} = −400 μA	All
V_{OH1}	Output High Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)	2.4			V	I_{OH} = −100 μA	All
V_{OH2}	Output High Voltage (All Other Outputs)	2.4			V	I_{OH} = −40 μA	All

D.C. CHARACTERISTICS T_A = 0°C to +70°C; V_{CC} = V_{DD} = 5V ±10%; V_{SS} = 0V (Continued)

Symbol	Parameter	Limits			Unit	Test Conditions	Device
		Min	Typ	Max			
I_{L1}	Leakage Current (T1, $\overline{INT}$)			±10	μA	$V_{SS} \le V_{IN} \le V_{CC}$	All
I_{LI1}	Input Leakage Current (P10–P17, P20–P27, EA, $\overline{SS}$)			−500	μA	$V_{SS} + 0.45 \le V_{IN} \le V_{CC}$	All
I_{LI2}	Input Leakage Current $\overline{RESET}$	−10		−300	μA	$V_{SS} \le V_{IN} \le 3.8$	All
I_{L0}	Leakage Current (BUS, T0) (High Impedance State)			±10	μA	$V_{SS} \le V_{IN} \le V_{CC}$	All
I_{DD}	V_{DD} Supply Current (RAM Standby)		3	5	mA		8048AH 8035AHL
			4	7	mA		8049AH 8039AHL
			5	10	mA		8050AH 8040AHL
$I_{DD} + I_{CC}$	Total Supply Current*		30	65	mA		8048AH 8035AHL
			35	70	mA		8049AH 8039AHL
			40	80	mA		8050AH 8040AHL
			30	100	mA		P8748H
			50	110	mA		P8749H
V_{DD}	RAM Standby Voltage	2.2		5.5	V	Standby Mode Reset $\le V_{IL1}$	8048AH 8035AH
		2.2		5.5	V		8049AH 8039AH
		2.2		5.5	V		8050AH 8040AHL

*I_{CC} + I_{DD} are measured with all outputs in their high impedance state; $\overline{RESET}$ low; 11 MHz crystal applied; $\overline{INT}$, $\overline{SS}$, and EA floating.

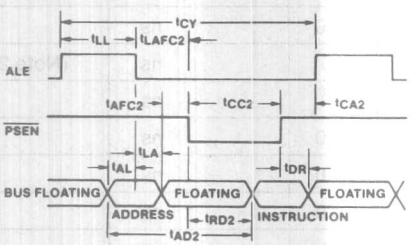
A.C. CHARACTERISTICS $T_A = 0°C$ to $+70°C$; $V_{CC} = V_{DD} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	f (t) (Note 3)	11 MHz		Unit	Conditions (Note 1)
			Min	Max		
t	Clock Period	1/xtal freq	90.9	1000	ns	(Note 3)
t_{LL}	ALE Pulse Width	3.5t−170	150		ns	
t_{AL}	Addr Setup to ALE	2t−110	70		ns	(Note 2)
t_{LA}	Addr Hold from ALE	t−40	50		ns	
t_{CC1}	Control Pulse Width ($\overline{RD}$, $\overline{WR}$)	7.5t−200	480		ns	
t_{CC2}	Control Pulse Width ($\overline{PSEN}$)	6t−200	350		ns	
t_{DW}	Data Setup before $\overline{WR}$	6.5t−200	390		ns	
t_{WD}	Data Hold after $\overline{WR}$	t−50	40		ns	
t_{DR}	Data Hold ($\overline{RD}$, $\overline{PSEN}$)	1.5t−30	0	110	ns	
t_{RD1}	$\overline{RD}$ to Data in	6t−170		375	ns	
t_{RD2}	$\overline{PSEN}$ to Data in	4.5t−170		240	ns	
t_{AW}	Addr Setup to $\overline{WR}$	5t−150	300		ns	
t_{AD1}	Addr Setup to Data ($\overline{RD}$)	10.5t−220		730	ns	
t_{AD2}	Addr Setup to Data ($\overline{PSEN}$)	7.5t−200		460	ns	
t_{AFC1}	Addr Float to $\overline{RD}$, $\overline{WR}$	2t−40	140		ns	(Note 2)
t_{AFC2}	Addr Float to $\overline{PSEN}$	0.5t−40	10		ns	(Note 2)
t_{LAFC1}	ALE to Control ($\overline{RD}$, $\overline{WR}$)	3t−75	200		ns	
t_{LAFC2}	ALE to Control ($\overline{PSEN}$)	1.5t−75	60		ns	
t_{CA1}	Control to ALE ($\overline{RD}$, $\overline{WR}$, PROG)	t−65	25		ns	
t_{CA2}	Control to ALE ($\overline{PSEN}$)	4t−70	290		ns	
t_{CP}	Port Control Setup to PROG	1.5t−80	50		ns	
t_{PC}	Port Control Hold to PROG	4t−260	100		ns	
t_{PR}	PROG to P2 Input Valid	8.5t−120		650	ns	
t_{PF}	Input Data Hold from PROG	1.5t	0	140	ns	
t_{DP}	Output Data Setup	6t−290	250		ns	
t_{PD}	Output Data Hold	1.5t−90	40		ns	
t_{PP}	PROG Pulse Width	10.5t−250	700		ns	
t_{PL}	Port 2 I/O Setup to ALE	4t−200	160		ns	
t_{LP}	Port 2 I/O Hold to ALE	0.5t−30	15		ns	
t_{PV}	Port Output from ALE	4.5t+100		5.0	ns	
t_{0PRR}	T0 Rep Rate	3t	270		ns	
t_{CY}	Cycle Time	15t	1.36	15.0	μs	

NOTES:
1. Control outputs: $C_L = 80$ pF. BUS Outputs: $C_L = 150$ pF.
2. BUS High Impedance Load 20 pF
3. f(t) assumes 50% duty cycle on X1, X2. Max clock period is for a 1 MHz crystal input.

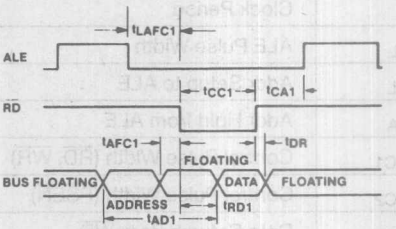
WAVEFORMS

INSTRUCTION FETCH FROM PROGRAM MEMORY



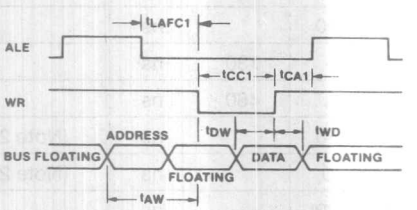
270053-4

READ FROM EXTERNAL DATA MEMORY



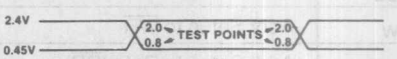
270053-5

WRITE TO EXTERNAL DATA MEMORY



270053-6

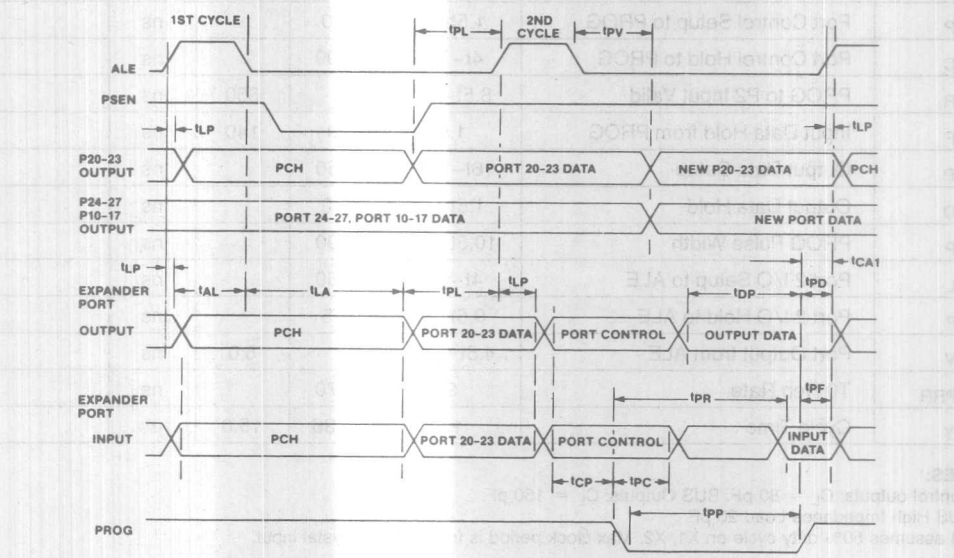
INPUT AND OUTPUT FOR A.C. TESTS



270053-7

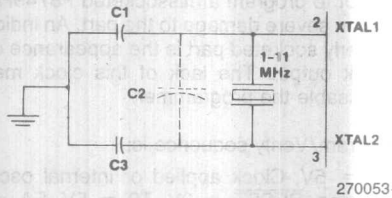
A.C. testing inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Output timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".

PORT 1/PORT 2 TIMING



270053-8

CRYSTAL OSCILLATOR MODE

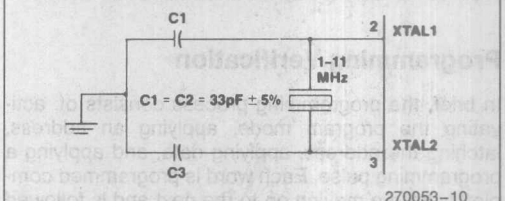


270053–9

C1 = 5 pF ±½ pF + (STRAY < 5 pF)
C2 = (CRYSTAL + STAY) < 3 pF
C3 = 20 pF ±1 pF + (STRAY < 5 pF)

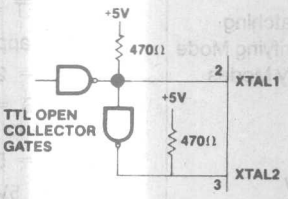
Crystal series resistance should be less than 30Ω at 11 MHz; less than 75Ω at 6 MHz; less than 180Ω at 3.6 MHz.

CERAMIC RESONATOR MODE



270053–10

DRIVING FROM EXTERNAL SOURCE



270053–11

For XTAL1 and XTAL2 define "high" as voltages above 1.6V and "low" as voltages below 1.6V. The duty cycle requirements for externally driving XTAL1 and XTAL2 using the circuits shown above are as follows: XTAL1 must be high 35–65% of the period and XTAL2 must be high 35–65% of the period. Rise and fall times must be faster than 20 ns.

PROGRAMMING AND VERIFYING THE P8749H/48H PROGRAMMABLE ROM

Programming Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pin	Function
XTAL1	Clock Input (3 to 4.0 MHz)
XTAL2	
$\overline{\text{RESET}}$	Initialization and Address Latching
T0	Selection of Program or Verifying Mode
EA	Activation of Program/Verify Modes
BUS	Address and Data Input
	Data Output During Verify
P20–P22	Address Input
V_{DD}	Programming Power Supply
PROG	Program Pulse Input

WARNING:

An attempt to program a missocketed P8749H/48H will result in severe damage to the part. An indication of a properly socketed part is the appearance of the ALE clock output. The lack of this clock may be used to disable the programmer.

The Program/Verify sequence is:

1. V_{DD} = 5V, Clock applied or internal oscillator operating, $\overline{\text{RESET}}$ = 0V, T0 = 5V, EA = 5V, BUS and PROG floating. P10 and P11 must be tied to ground.
2. Insert P8749H/48H in programming socket
3. T0 = 0V (select program mode)
4. EA = 18V (activate program mode)
5. Address applied to BUS and P20–22
6. $\overline{\text{RESET}}$ = 5V (latch address)
7. Data applied to BUS
8. V_{DD} = 21V (programming power)
9. PROG = V_{CC} or float followed by one 50 ms pulse to 18V
10. V_{DD} = 5V
11. T0 = 5V (verify mode)
12. Read and verify data on BUS
13. T0 = 0V
14. $\overline{\text{RESET}}$ = 0V and repeat from step 5
15. Programmer should be at conditions of step 1 when P8749H/48H is removed from socket.

NOTE:
Once programmed the P8749H/48H cannot be erased.

A.C. TIMING SPECIFICATION FOR PROGRAMMING P8748H/P8749H ONLY

T_A = 25°C ±5°C; V_{CC} = 5V ±5%; V_{DD} = 21 ±0.5V

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AW}	Address Setup Time to $\overline{RESET}$	$4t_{CY}$			
t_{WA}	Address Hold Time After $\overline{RESET}$	$4t_{CY}$			
t_{DW}	Data in Setup Time to PROG	$4t_{CY}$			
t_{WD}	Data in Hold Time After PROG	$4t_{CY}$			
t_{PH}	$\overline{RESET}$ Hold Time to Verify	$4t_{CY}$			
t_{VDDW}	V_{DD} Hold Time Before PROG	0	1.0	ms	
t_{VDDH}	V_{DD} Hold Time After PROG	0	1.0	ms	
t_{PW}	Program Pulse Width	50	60	ms	
t_{TW}	T0 Setup Time for Program Mode	$4t_{CY}$			
t_{WT}	T0 Hold Time After Program Mode	$4t_{CY}$			
t_{DO}	T0 to Data Out Delay		$4t_{CY}$		
t_{WW}	$\overline{RESET}$ Pulse Width to Latch Address	$4t_{CY}$			
t_r, t_f	V_{DD} and PROG Rise and Fall Times	0.5	100	µs	
t_{CY}	CPU Operation Cycle Time	3.75	5	µs	
t_{RE}	$\overline{RESET}$ Setup Time before EA	$4t_{CY}$			

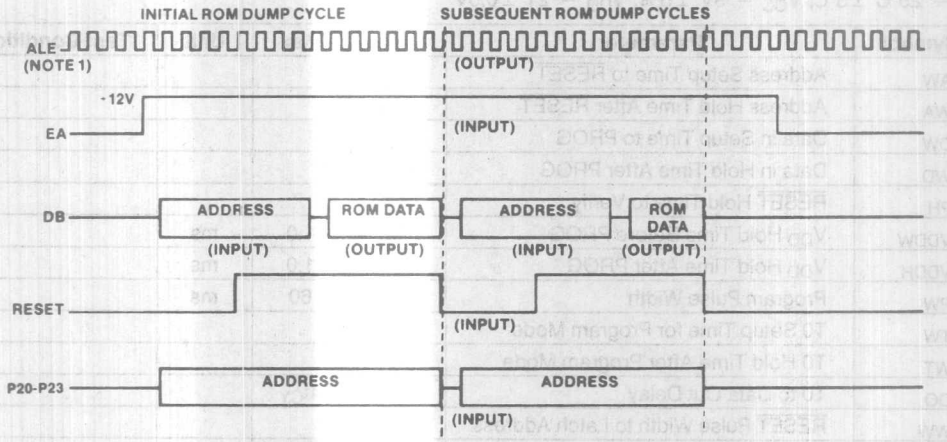
NOTE:
If Test 0 is high, t_{DO} can be triggered by $\overline{RESET}$.

D.C. CHARACTERISTICS FOR PROGRAMMING P8748H/P8749H ONLY

T_A = 25°C ±5°C; V_{CC} = 5V ±5%; V_{DD} = 21 ±0.5V

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{DDH}	V_{DD} Program Voltage High Level	20.5	21.5	V	
V_{DDL}	V_{DD} Voltage Low Level	4.75	5.25	V	
V_{PH}	PROG Program Voltage High Level	17.5	18.5	V	
V_{PL}	PROG Voltage Low Level	4.0	V_{CC}	V	
V_{EAH}	EA Program or Verify Voltage High Level	17.5	18.5	V	
I_{DD}	V_{DD} High Voltage Supply Current		20.0	mA	
I_{PROG}	PROG High Voltage Supply Current		1.0	mA	
I_{EA}	EA High Voltage Supply Current		1.0	mA	

SUGGESTED ROM VERIFICATION ALGORITHM FOR ROM DEVICE ONLY



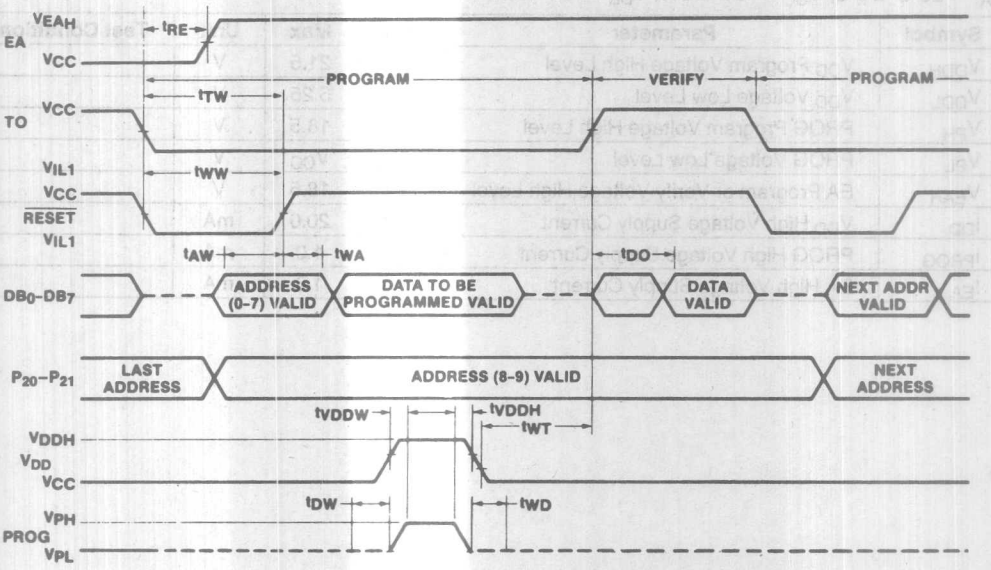
270053-12

	48H	49H	50H
A10	0	ADDR	ADDR
A11	0	0	ADDR

$V_{CC} = V_{DD} = +5V$
$V_{SS} = 0V$

NOTE:
ALE is function of X1, X2 inputs.

COMBINATION PROGRAM/VERIFY MODE (PROGRAMMABLE ROMS ONLY)



270053-13

D8748H/D8749H
HMOS-E SINGLE-COMPONENT 8-BIT MICROCOMPUTER

- **High Performance HMOS-E**
- **Interval Timer/Event Counter**
- **Two Single Level Interrupts**
- **Single 5-Volt Supply**
- **Over 96 Instructions; 90% Single Byte**
- **Compatible with 8080/8085 Peripherals**
- **Easily Expandable Memory and I/O**
- **Up to 1.35 μs Instruction Cycle; All Instructions 1 or 2 Cycles**

The Intel D8749H/D8748H are totally self-sufficient, 8-bit parallel computers fabricated on single silicon chips using Intel's advanced N-channel silicon gate HMOS-E process.

The family contains 27 I/O lines, an 8-bit timer/counter, on-chip RAM and on-board oscillator/clock circuits. For systems that require extra capability, the family can be expanded using MCS®-80/MCS®-85 peripherals.

These microcomputers are designed to be efficient controllers as well as arithmetic processors. They have extensive bit handling capability as well as facilities for both binary and BCD arithmetic. Efficient use of program memory results from an instruction set consisting mostly of single byte instructions and no instructions over 2 bytes in length.

Device	Internal Memory	
D8749H	2K x 8 EPROM	128 x 8 RAM
D8748H	1K x 8 EPROM	64 x 8 RAM

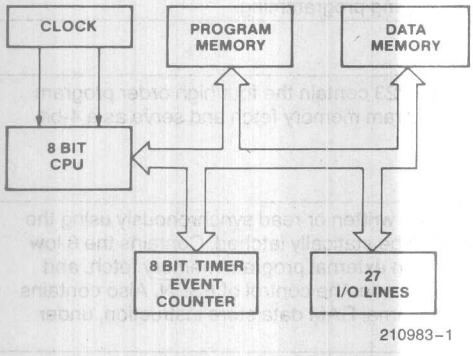


Figure 1. Block Diagram

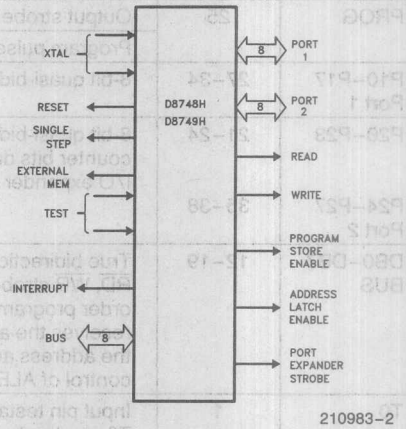


Figure 2. Logic Symbol

October 1987
Order Number: 210983-003

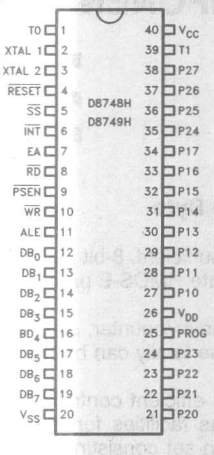


Figure 3. Pin Configuration

Table 1. Pin Description (40-Pin DIP)

Symbol	Pin No.	Function
V_{SS}	20	Circuit GND potential.
V_{DD}	26	+5V during normal operation.
		Programming power supply (+21V).
V_{CC}	40	Main power supply; +5V during operation and programming.
PROG	25	Output strobe for 8243 I/O expander.
		Program pulse (+18V) input pin during programming.
P10–P17 Port 1	27–34	8-bit quasi-bidirectional port.
P20–P23	21–24	8-bit quasi-bidirectional port. P20–P23 contain the four high order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for 8243.
P24–P27 Port 2	35–38	
DB0–DB7 BUS	12–19	True bidirectional port which can be written or read synchronously using the $\overline{RD}$, $\overline{WR}$ strobes. The port can also be statically latched. Contains the 8 low order program counter bits during an external program memory fetch, and receives the addressed instruction under the control of $\overline{PSEN}$. Also contains the address and data during an external RAM data store instruction, under control of ALE, $\overline{RD}$, and $\overline{WR}$.
T0	1	Input pin testable using the conditional transfer instructions JT0 and JNT0. T0 can be designated as a clock output using ENT0 CKL instruction.
		Used during programming.
T1	39	Input pin testable using the JT1, and JNT1 instructions. Can be designated the timer/counter input using the STRT CNT instruction.
$\overline{INT}$	6	Interrupt input. Initiates an interrupt if interrupt is enabled. Interrupt is disabled after a reset. Also testable with conditional jump instruction. (Active low) interrupt must remain low for at least 3 machine cycles for proper operation.
$\overline{RD}$	8	Output strobe activated during a BUS read. Can be used to enable data onto the bus from an external device. Used as a read strobe to external data memory. (Active low)

Table 1. Pin Description (40-Pin DIP) (Continued)

Symbol	Pin No.	Function
$\overline{RESET}$	4	Input which is used to initialize the processor. (Active low) (Non TTL V_{IH})
		Used during programming.
$\overline{WR}$	10	Output strobe during a bus write. (Active low) Used as write strobe to external data memory.
ALE	11	Address latch enable. This signal occurs once during each cycle and is useful as a clock output. The negative edge of ALE strobes address into external data and program memory.
$\overline{PSEN}$	9	Program store enable. This output occurs only during a fetch to external program memory. (Active low.)
$\overline{SS}$	5	Single step input can be used in conjunction with ALE to "single step" the processor through each instruction.
EA	7	External access input which forces all program memory fetches to reference external memory. Useful for emulation and debug. (Active high.)
		Used during (18V) programming.
XTAL1	2	One side of crystal input for internal oscillator. Also input for external source. (Non TTL V_{IH}.)
XTAL2	3	Other side of crystal input.

Table 2. Instruction Set

Mnemonic	Description	Bytes	Cycles
ACCUMULATOR			
ADD A, R	Add register to A	1	1
ADD A,@R	Add data memory to A	1	1
ADD A, #data	Add immediate to A	2	2
ADDC A, R	Add register with carry	1	1
ADDC A, @R	Add data memory with carry	1	1
ADDC A, #data	Add immediate with carry	2	2
ANL A, R	And register to A	1	1
ANL A, @R	And data memory to A	1	1
ANL A, #data	And immediate to A	2	2
ORL A, R	Or register to A	1	1
ORL A, @R	Or data memory to A	1	1
ORL A, #data	Or immediate to A	2	2
XRL A, R	Exclusive or register to A	1	1
XRL A, @R	Exclusive or data memory to A	1	1
XRL A, #data	Exclusive or immediate to A	2	2

Mnemonic	Description	Bytes	Cycles
ACCUMULATOR (Continued)			
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
INPUT/OUTPUT			
IN A, P	Input port to A	1	2
OUTL P, A	Output A to port	1	2
ANL P, #data	And immediate to port	2	2
ORL P, #data	Or immediate to port	2	2
INS A, BUS	Input BUS to A	1	2
OUTL BUS, A	Output A to BUS	1	2
ANL BUS, #data	And immediate to BUS	2	2
ORL BUS, #data	Or immediate to BUS	2	2
MOVD A, P	Input expander port to A	1	2

Table 2. Instruction Set (Continued)

Mnemonic	Description	Bytes	Cycles
INPUT/OUTPUT (Continued)			
MOVD P, A	Output A to expander port	1	2
ANLD P, A	And A to expander port	1	2
ORLD P, A	Or A to expander port	1	2
REGISTERS			
INC R	Increment register	1	1
INC @R	Increment data memory	1	1
DEC R	Decrement register	1	1
BRANCH			
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R, addr	Decrement register and skip	2	2
JC addr	Jump on carry = 1	2	2
JNC addr	Jump on carry = 0	2	2
JZ addr	Jump on A zero	2	2
JNZ addr	Jump on A not zero	2	2
JT0 addr	Jump on T0 = 1	2	2
JNT0 addr	Jump on T0 = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 = 1	2	2
JF1 addr	Jump on F1 = 1	2	2
JTF addr	Jump on timer flag	2	2
JNI addr	Jump on INT = 0	2	2
JBb addr	Jump on accumulator bit	2	2
SUBROUTINE			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
FLAGS			
CLR C	Clear carry	1	1
CPL C	Complement carry	1	1
CLR F0	Clear flag 0	1	1
CPL F0	Complement flag 0	1	1
CLR F1	Clear flag 1	1	1
CPL F1	Complement flag 1	1	1
DATA MOVES			
MOV A, R	Move register to A	1	1
MOV A, @R	Move data memory to A	1	1
MOV A, #data	Move immediate to A	2	2

Mnemonic	Description	Bytes	Cycles
DATA MOVES (Continued)			
MOV R, A	Move A to register	1	1
MOV @R, A	Move A to data memory	1	1
MOV R, #data	Move immediate to register	2	2
MOV @R, #data	Move immediate to data memory	2	2
MOV A, PSW	Move PSW to A	1	1
MOV PSW, A	Move A to PSW	1	1
XCH A, R	Exchange A and register	1	1
XCH A, @R	Exchange A and data memory	1	1
XCHD A, @R	Exchange nibble of A and register	1	1
MOVX A, @R	Move external data memory to A	1	2
MOVX @R, A	Move A to external data memory	1	2
MOVP A, @A	Move to A from current page	1	2
MOVP3 A, @A	Move to A from page 3	1	2
TIMER/COUNTER			
MOV A, T	Read timer/counter	1	1
MOV T, A	Load timer/counter	1	1
STRT T	Start timer	1	1
STRT CNT	Start counter	1	1
STOP TCNT	Stop timer/counter	1	1
EN TCNTI	Enable timer/counter interrupt	1	1
DIS TCNTI	Disable timer/counter interrupt	1	1
CONTROL			
EN I	Enable external interrupt	1	1
DIS I	Disable external interrupt	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
SEL MB0	Select memory bank 0	1	1
SEL MB1	Select memory bank 1	1	1
ENT0 CLK	Enable clock output on T0	1	1
NOP	No operation	1	1

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage On Any Pin With Respect to Ground . . . . . −0.5V to +7V
Power Dissipation . . . . . 1.0 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS $T_A = 0°C$ to $+70°C$; $V_{CC} = V_{DD} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	Limits			Unit	Test Conditions	Device
		Min	Typ	Max			
V_{IL}	Input Low Voltage (All Except RESET, X1, X2)	−0.5		0.8	V		All
V_{IL1}	Input Low Voltage (RESET, X1, X2)	−0.5		0.6	V		All
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, RESET	2.0		V_{CC}	V		All
V_{IH1}	Input High Voltage (X1, X2, RESET)	3.8		V_{CC}	V		All
V_{OL}	Output Low Voltage (BUS)			0.45	V	$I_{OL} = 2.0$ mA	All
V_{OL1}	Output Low Voltage (RD, WR, PSEN, ALE)			0.45	V	$I_{OL} = 1.8$ mA	All
V_{OL2}	Output Low Voltage (PROG)			0.45	V	$I_{OL} = 1.0$ mA	All
V_{OL3}	Output Low Voltage (All Other Outputs)			0.45	V	$I_{OL} = 1.6$ mA	All
V_{OH}	Output High Voltage (BUS)	2.4			V	$I_{OH} = -400\ \mu A$	All
V_{OH1}	Output High Voltage (RD, WR, PSEN, ALE)	2.4			V	$I_{OH} = -100\ \mu A$	All
V_{OH2}	Output High Voltage (All Other Outputs)	2.4			V	$I_{OH} = -40\ \mu A$	All
I_{L1}	Leakage Current (T1, INT)			±10	μA	$V_{SS} \le V_{IN} \le V_{CC}$	All
I_{LI1}	Input Leakage Current (P10–P17, P20–P27, EA, SS)			−500	μA	$V_{SS} + 0.45 \le V_{IN} \le V_{CC}$	All
I_{LI2}	Input Leakage Current RESET	−10		−300	μA	$V_{SS} \le V_{IN} \le 3.8V$	All
I_{L0}	Leakage Current (BUS, T0) (High Impedance State)			±10	μA	$V_{SS} \le V_{IN} \le V_{CC}$	All
$I_{DD} + I_{CC}$	Total Supply Current*		80	100	mA		8748H
			95	110	mA		8749H

NOTE:
*$I_{CC} + I_{DD}$ is measured with all outputs disconnected; SS, RESET, and INT equal to V_{CC}; EA equal to V_{SS}.

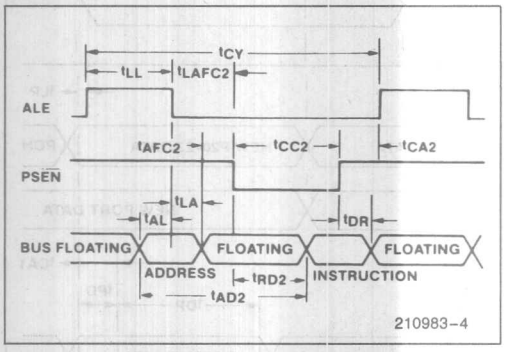
A.C. CHARACTERISTICS $T_A = 0°C$ to $+70°C$; $V_{CC} = V_{DD} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	f(t) (Note 3)	11 MHz		Unit	Conditions (Note 1)
			Min	Max		
t	Clock Period	1/xtal freq	90.9	1000	ns	(Note 3)
t_{LL}	ALE Pulse Width	3.5t − 170	150		ns	
t_{AL}	Addr Setup to ALE	2t − 110	70		ns	(Note 2)
t_{LA}	Addr Hold from ALE	t − 40	50		ns	
t_{CC1}	Control Pulse Width ($\overline{RD}$, $\overline{WR}$)	7.5t − 200	480		ns	
t_{CC2}	Control Pulse Width ($\overline{PSEN}$)	6t − 200	350		ns	
t_{DW}	Data Setup before $\overline{WR}$	6.5t − 200	390		ns	
t_{WD}	Data Hold after $\overline{WR}$	t − 50	40		ns	
t_{DR}	Data Hold ($\overline{RD}$, $\overline{PSEN}$)	1.5t − 30	0	110	ns	
t_{RD1}	$\overline{RD}$ to Data In	6t − 170		375	ns	
t_{RD2}	$\overline{PSEN}$ to Data In	4.5t − 170		240	ns	
t_{AW}	Addr Setup to $\overline{WR}$	5t − 150	300		ns	
t_{AD1}	Addr Setup to Data ($\overline{RD}$)	10.5t − 220		730	ns	
t_{AD2}	Addr Setup to Data ($\overline{PSEN}$)	7.5t − 200		460	ns	
t_{AFC1}	Addr Float to $\overline{RD}$, $\overline{WR}$	2t − 40	140		ns	(Note 2)
t_{AFC2}	Addr Float to $\overline{PSEN}$	0.5t − 40	10		ns	(Note 2)
t_{LAFC1}	ALE to Control ($\overline{RD}$, $\overline{WR}$)	3t − 75	200		ns	
t_{LAFC2}	ALE to Control ($\overline{PSEN}$)	1.5t − 75	60		ns	
t_{CA1}	Control to ALE ($\overline{RD}$, $\overline{WR}$, PROG)	t − 65	25		ns	
t_{CA2}	Control to ALE ($\overline{PSEN}$)	4t − 70	290		ns	
t_{CP}	Port Control Setup to PROG	1.5t − 80	50		ns	
t_{PC}	Port Control Hold to PROG	4t − 260	100		ns	
t_{PR}	PROG to P2 Input Valid	8.5t − 120		650	ns	
t_{PF}	Input Data Hold from PROG	1.5t	0	140	ns	
t_{DP}	Output Data Setup	6t − 290	250		ns	
t_{PD}	Output Data Hold	1.5t − 90	40		ns	
t_{PP}	PROG Pulse Width	10.5t − 250	700		ns	
t_{PL}	Port 2 I/O Setup to ALE	4t − 200	160		ns	
t_{LP}	Port 2 I/O Hold to ALE	0.5t − 30	15		ns	
t_{PV}	Port Output from ALE	4.5t + 100		510	ns	
t_{0PRR}	T0 Rep Rate	3t	270		ns	
t_{CY}	Cycle Time	15t	1.36	15.0	μs	

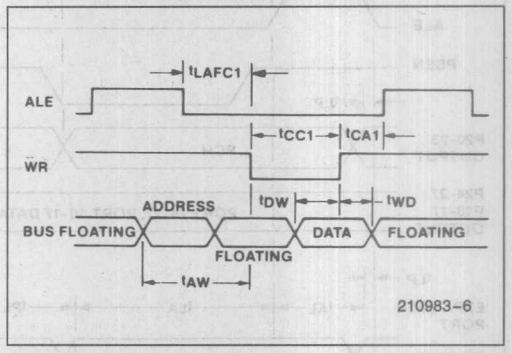
NOTES:
1. Control outputs CL = 80 pF; BUS outputs CL = 150 pF.
2. BUS High Impedance Load 20 pF.
3. f(t) assumes 50% duty cycle on X1, X2. Max clock period is for a 1 MHz crystal input.

WAVEFORMS

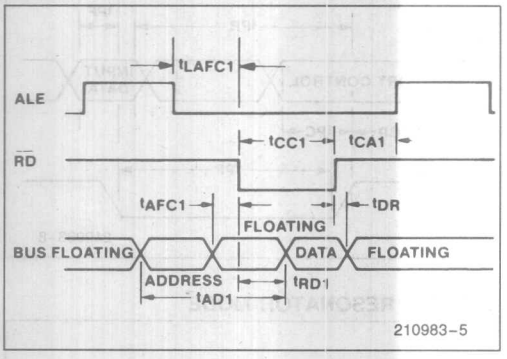
INSTRUCTION FETCH FROM PROGRAM MEMORY



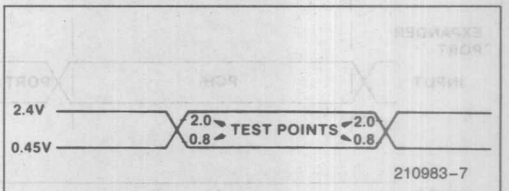
WRITE TO EXTERNAL DATA MEMORY



READ FROM EXTERNAL DATA MEMORY

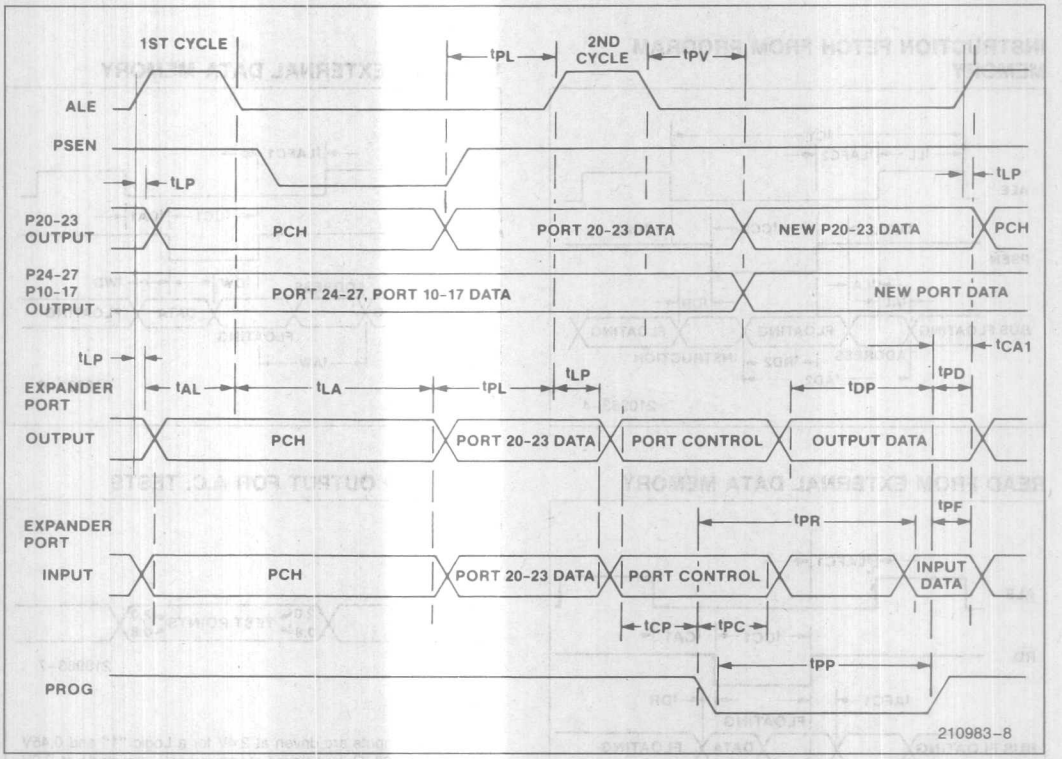


INPUT AND OUTPUT FOR A.C. TESTS

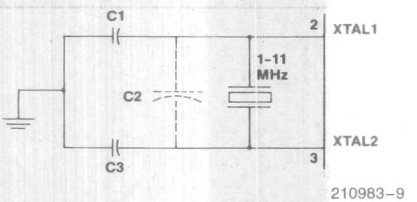


A.C. testing inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Output timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0."

PORT 1/PORT 2 TIMING



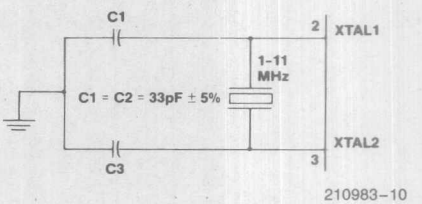
CRYSTAL OSCILLATOR MODE



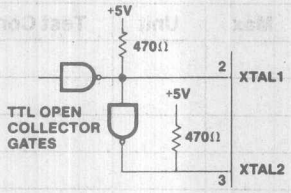
C1 = 5 pF ±½ pF + (STRAY < 5 pF)
C2 = (CRYSTAL + STRAY) < 8 pF
C3 = 20 pF ±1 pF (STRAY < 5 pF)

Crystal series resistance should be less than 30Ω at 11 MHz; less than 75Ω at 6 MHz; less than 180Ω at 3.6 MHz.

CERAMIC RESONATOR MODE



DRIVING FROM EXTERNAL SOURCE



210983-11

For XTAL1 and XTAL2 define "high" as voltages above 1.6V and "low" as voltages below 1.6V. The duty cycle requirements for externally driving XTAL1 and XTAL2 using the circuit shown above are as follows: XTAL1 must be high 35–65% of the period and XTAL2 must be high 36–65% of the period. Rise and fall times must be faster than 20 ns.

PROGRAMMING, VERIFYING AND ERASING THE 8749H (8748H) EPROM

Programming Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pin	Function
XTAL 1	Clock Input (3 to 4.0 MHz)
XTAL 2	
RESET	Initialization and Address Latching
TEST 0	Selection of Program or Verify Mode
EA	Activation of Program/Verify Modes
BUS	Address and Data Input
	Data Output During Verify
P20–P22	Address Input
V_{DD}	Programming Power Supply
PROG	Program Pulse Input

WARNING

An attempt to program a missocketed 8749H (8748H) will result in severe damage to the part. An indication of a properly socketed part is the appearance of the ALE clock output. The lack of this clock may be used to disable the programmer.

The Program/Verify sequence is:

1) V_{DD} = 5V, Clock applied or internal oscillator operating. RESET = 0V, TEST 0 = 5V, EA = 5V, BUS and PROG floating. P10 and P11 must be tied to ground.
2) Insert 8749H (8748H) in programming socket.
3) TEST 0 = 0V (select program mode)
4) EA = 18V (activate program mode)
5) Address applied to BUS and P20–22
6) RESET = 5V (latch address)
7) Data applied to BUS
8) V_{DD} = 21V (programming power)
9) PROG = V_{CC} or float followed by one 50 ms pulse to 18V
10) V_{DD} = 5V
11) TEST 0 = 5V (verify mode)
12) Read and verify data on BUS
13) TEST 0 = 0V
14) RESET = 0V and repeat from step 5
15) Programmer should be at conditions of step 1 when 8749H (8748H) is removed from socket.

A.C. TIMING SPECIFICATION FOR PROGRAMMING 8748H/8749H

T_A = 25°C ±5°C; V_{CC} = 5V ±5%; V_{DD} = 21V ±0.5V

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AW}	Address Setup Time to $\overline{RESET}$ ↑	$4t_{CY}$			
t_{WA}	Address Hold Time after $\overline{RESET}$ ↑	$4t_{CY}$			
t_{DW}	Data in Setup Time to PROG ↑	$4t_{CY}$			
t_{WD}	Data in Hold Time after PROG ↓	$4t_{CY}$			
t_{PH}	$\overline{RESET}$ Hold Time to Verify	$4t_{CY}$			
t_{VDDW}	V_{DD} Hold Time before PROG ↑	0	1.0	ms	
t_{VDDH}	V_{DD} Hold Time after PROG ↓	0	1.0	ms	
t_{PW}	Program Pulse Width	50	60	ms	
t_{TW}	TEST 0 Setup Time for Program Mode	$4t_{CY}$			
t_{WT}	TEST 0 Hold Time after Program Mode	$4t_{CY}$			
t_{DO}	TEST 0 to Data Out Delay		$4t_{CY}$		
t_{WW}	$\overline{RESET}$ Pulse Width to Latch Address	$4t_{CY}$			
t_r, t_f	V_{DD} and PROG Rise and Fall Times	0.5	100	μs	
t_{CY}	CPU Operation Cycle Time	3.75	5	μs	
t_{RE}	$\overline{RESET}$ Setup Time before EA ↑	$4t_{CY}$			

NOTE:
If TEST 0 is high, t_{DO} can be triggered by $\overline{RESET}$ ↑.

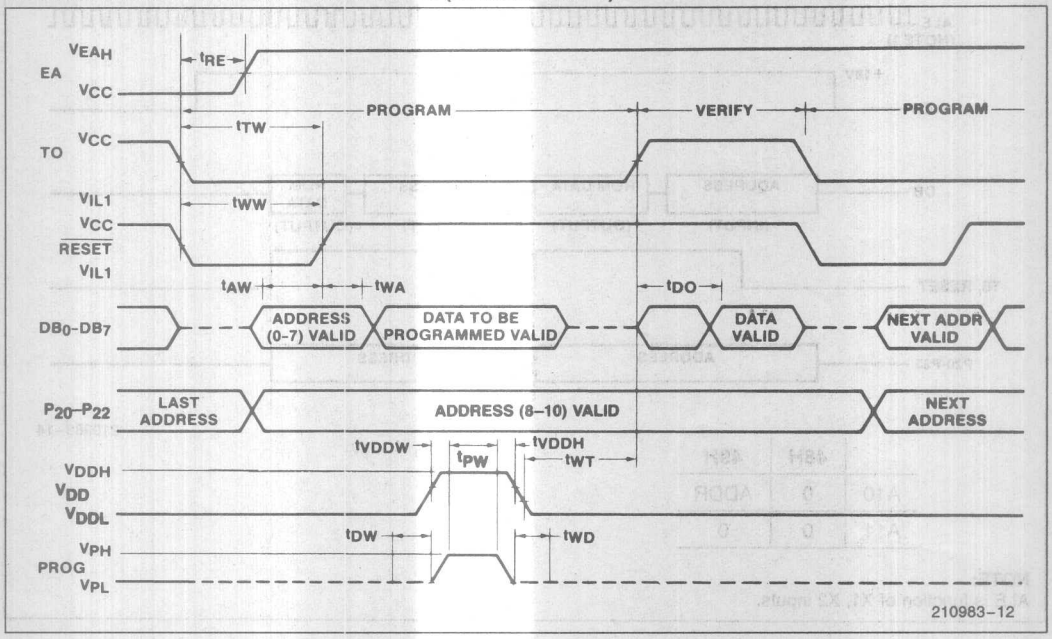
D.C. SPECIFICATION FOR PROGRAMMING 8748H/8749H

T_A = 25°C ±5°C; V_{CC} = 5V ±5%; V_{DD} = 21V ±0.5V

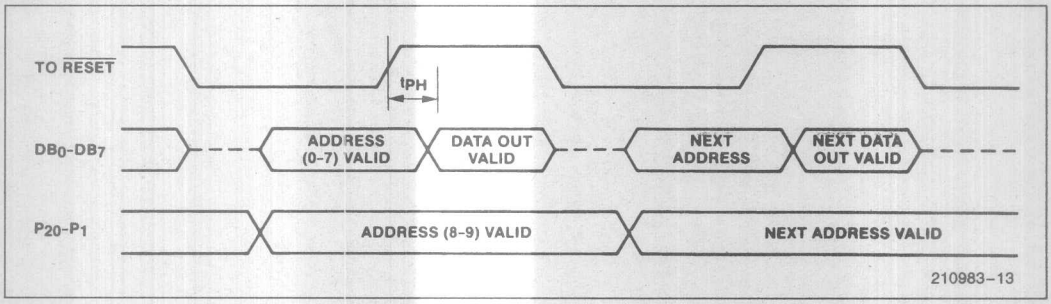
Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{DDH}	V_{DD} Program Voltage High Level	20.5	21.5	V	
V_{DDL}	V_{DD} Voltage Low Level	4.75	5.25	V	
V_{PH}	PROG Program Voltage High Level	17.5	18.5	V	
V_{PL}	PROG Voltage Low Level	4.0	V_{CC}	V	
V_{EAH}	EA Program or Verify Voltage High Level	17.5	18.5	V	
I_{DD}	V_{DD} High Voltage Supply Current		20.0	mA	
I_{PROG}	PROG High Voltage Supply Current		1.0	mA	
I_{EA}	EA High Voltage Supply Current		1.0	mA	

WAVEFORMS

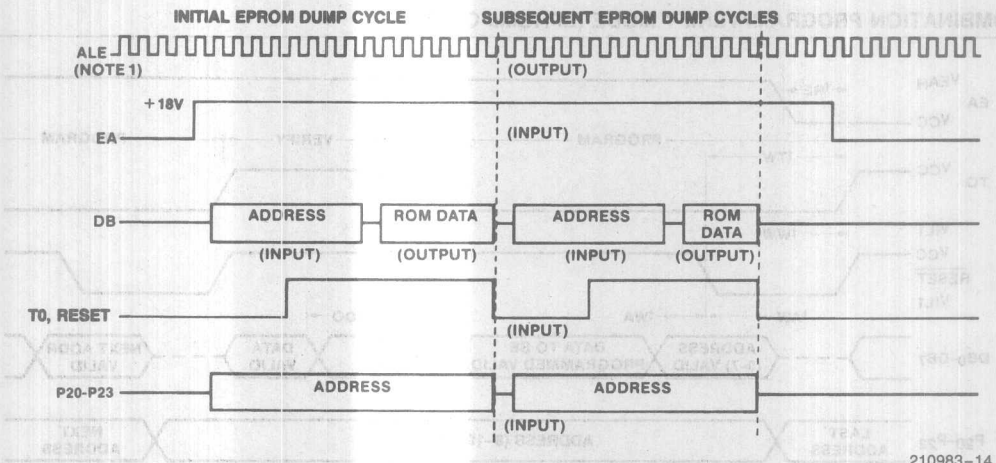
COMBINATION PROGRAM/VERIFY MODE (EPROMs ONLY)



VERIFY MODE



SUGGESTED EPROM VERIFICATION ALGORITHM FOR HMOS-E DEVICE ONLY



	48H	49H
A10	0	ADDR
A11	0	0

$V_{CC} = V_{DD} = +5V$
$V_{SS} = 0V$

NOTE:
ALE is function of X1, X2 inputs.

MCS®-48
EXPRESS

- 0°C to 70°C Operation
- −40°C to +85°C Operation
- 168 Hr. Burn-In
- 8048AH/8035AHL
- 8049AH/8039AHL
- 8050AH/8040AHL
- 8748H
- 8243
- 8749H

The new Intel EXPRESS family of single-component 8-bit microcomputers offers enhanced processing options to the familiar 8048AH/8035AHL, 8748H, 8049AH/8039AHL, 8749H, 8050AH/8040AHL Intel components. These EXPRESS products are designed to meet the needs of those applications whose operating requirements exceed commercial standards, but fall short of military conditions.

The EXPRESS options include the commercial standard and −40°C to +85°C operation with or without 168 ±8 hours of dynamic burn-in at 125°C per MIL-STD-883, method 1015. Figure 1 summarizes the option marking designators and package selections.

For a complete description of 8048AH/8035AHL, 8748H, 8049AH/8309AHL, 8749H, 8040AHL and 8050AH features and operating characteristics, refer to the respective standard commercial grade data sheet. This document highlights only the electrical specifications which differ from the respective commercial part.

Temp Range °C	0–70	−40–+85	0–70	−40–+85
Burn In	0 Hrs	0 Hrs	168 Hrs	168 Hrs
	P8048AH	TP8048AH	QP8048AH	LP8048AH
	D8048AH	TD8048AH	QD8048AH	LD8048AH
	D8748H	TD8748H	QD8748H	LD8748H
	P8035AHL	TP8035AHL	QP8035AHL	LP8035AHL
	D8035AHL	TD8035AHL	QD8035AHL	LD8035AHL
	P8049AH	TP8049AH	QP8049AH	LP8049AH
	D8049AH	TD8049AH	QD8049AH	LD8049AH
	D8749H	TD8749AH	QD8749H	LD8749AH
	P8039AHL	TP8039AHL	QP8039AHL	LP8039AHL
	D8039AHL	TD8039AHL	QD8039AHL	LD8039AHL
	P8050AH	TP8050AH	QP8050AH	LP8050AH
	D8050AH	TD8050AH	QD8050AH	LD8050AH
	P8040AHL	TP8040AHL	QP8040AHL	LP8040AHL
	D8040AHL	TD8040AHL	QD8040AHL	LD8040AHL
	P8243	TP8243	QP8243	—
	D8243	TD8243	QD8243	LD8243

* Commercial Grade
P Plastic Package
D Cerdip Package

September 1987
Order Number: 270225-002

Extended Temperature Electrical Specification Deviations*

TP8048AH/TP8035AHL/LP8048AH/LP8035AHL
TD8048AH/TD8035AHL/LD8048AH/LD8035AHL

D.C. CHARACTERISTICS $T_A = -40°C$ to $+85°C$; $V_{CC} = V_{DD} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)	2.2		V_{CC}	V	
I_{DD}	V_{DD} Supply Current		4	8	mA	
$I_{DD} + I_{CC}$	Total Supply Current		40	80	mA	

TP8049AH/TP8039AHL/LP8049AH/LP8039AHL
TD8049AH/TD8039AHL/LD8049AH/LD8039AHL

D.C. CHARACTERISTICS $T_A = -40°C$ to $+85°C$; $V_{CC} = V_{DD} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)	2.2		V_{CC}	V	
I_{DD}	V_{DD} Supply Current		5	10	mA	
$I_{DD} + I_{CC}$	Total Supply Current		50	100	mA	

TP8050AH/TP8040AHL/LP8050AHL/LP8040AHL
TD8050AH/TD8040AHL/LD8050AH/LD8040AHL

D.C. CHARACTERISTICS $T_A = -40°C$ to $+85°C$; $V_{CC} = V_{DD} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)	2.2		V_{CC}	V	
I_{DD}	V_{DD} Supply Current		10	20	mA	
$I_{DD} + I_{CC}$	Total Supply Current		75	120	mA	

*Extended Temperature Electrical Specification Deviations**

TD8748H/LD8748H

D.C. CHARACTERISTICS $T_A = -40°C$ to $+85°C$; $V_{CC} = V_{DD} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)	2.2		V_{CC}	V	
$I_{DD} + I_{CC}$	Total Supply Current		50	130	mA	

TD8749H/LD8749H

D.C. CHARACTERISTICS $T_A = -40°C$ to $+85°C$; $V_{CC} = V_{DD} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)	2.2		V_{CC}	V	
$I_{DD} + I_{CC}$	Total Supply Current		75	150	mA	

TP8743/TD8243/LD8243

D.C. CHARACTERISTICS $T_A = -40°C$ to $+85°C$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
I_{CC}	V_{CC} Supply Current		15	25	mA	

*Refer to individual commercial grade data sheet for complete operating characteristics.

MCS®-48 INDEX

8

8243 Expander, 2-4
8243 Port Characteristics, 2-5

A

Accumulator, 1-1
Addressing Beyond 2K, 2-1
Addressing External Data Memory, 2-4
ALE, 1-17, 2-9
ALU, 1-1
Arithmetic Logic Unit, 1-1

B

Bus, 1-5, 1-17, 2-9

C

Clock Circuits, 1-9
Conditional Branches, 1-6
Control Signals, 2-8
Counter, 1-7
Cycle Counter, 1-10

D

Data Memory, 1-3

E

EA, 1-15
Erasing EPROM, 1-18
Erasure Characteristics, 1-18
Event Counter, 1-9
Expansion of Data Memory, 2-3
Expansion of I/O, 2-4
Expansion of Program Memory, 2-1
Extended Addressing, 2-1
External Access Mode, 1-15
External Data Memory Addressing, 2-4
External Instruction Fetch, 2-1

I

I/O Expander Device (8243), 2-4
I/O Expansion, 2-4, 2-5
I/O Port Characteristics, 2-5
I/O Port Restore, 2-2
Instruction Decoder, 1-1
Instruction Fetch (External), 2-1
INT, 1-17
Interrupt, 1-5, 1-7
Interrupt Routines, 2-2
Interrupt Timing, 1-7

M

Memory Bank Switch, 2-1
Memory Bank Switching, 2-8
Memory Expansion, 2-5
Multi-Chip Systems, 2-7

O

Oscillator, 1-9

P

Pin Description, 1-16
Port 1, 1-5, 1-17
Port 2, 1-5, 2-9
Port Characteristics, 2-9
Power Down, 1-13
PROG, 1-17, 2-9
Program Counter, 1-5
Program Memory, 1-1
Program Status Word, 1-6
Programming EPROM, 1-18
PSEN, 2-9
PSW, 1-6

R

RD, 1-17, 2-9
Read Cycle, 2-3
Reset, 1-10, 1-17
Restoring I/O Ports, 2-2

S

Single Step, 1-11, 1-14
Stack, 1-5
State Counter, 1-10
Sync Mode, 1-15

T

T0, 1-5, 1-17
T1, 1-5, 1-17
Test Inputs, 1-5
Timer, 1-7, 1-9
Timing, 1-13
Timing Circuits, 1-9

V

V_{CC}, 1-17
V_{DD}, 1-17
Verifying EPROM, 1-18
V_{SS}, 1-17

W

WR, 1-17, 2-9
Write Cycle, 2-3

Overview

5 MCS®-51 Architecture Overview

ARCHITECTURAL OVERVIEW OF THE MCS®-51 FAMILY OF MICROCONTROLLERS

MEMBERS OF THE FAMILY

The MCS®-51 family of microcontrollers consists of the devices listed in Table 1. The basic architectural structure of these devices is shown in Figure 1.

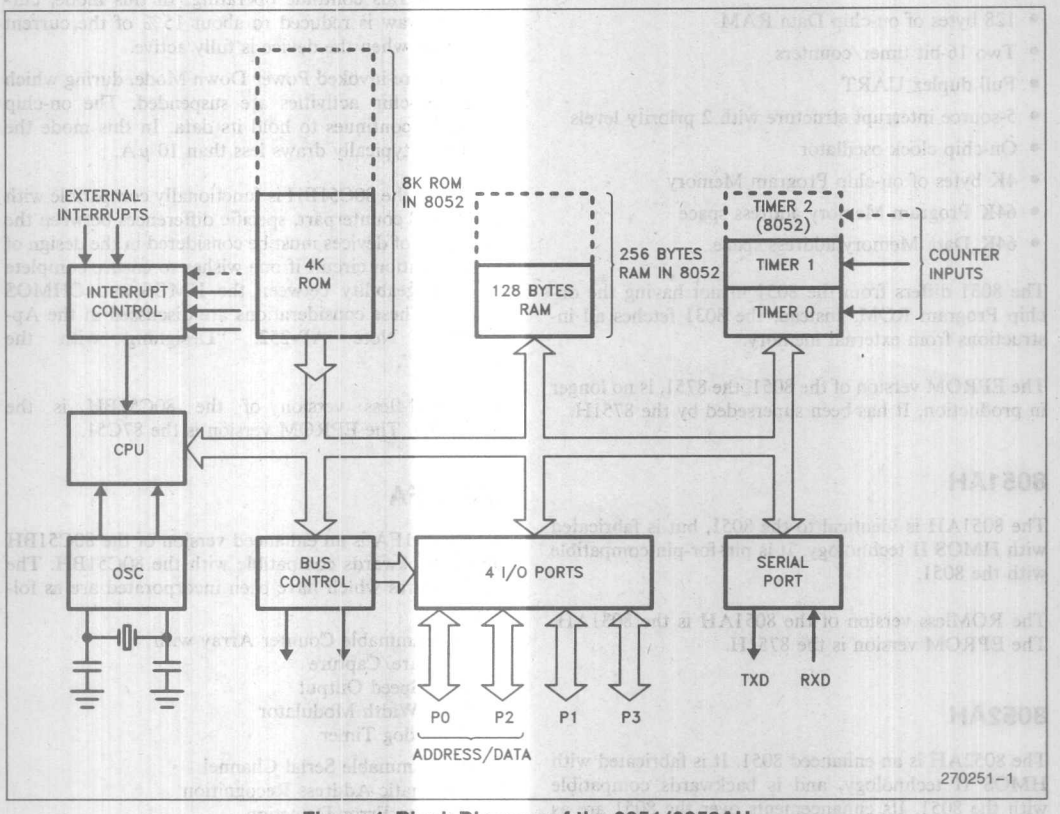


Figure 1. Block Diagram of the 8051/8052AH

Table 1. The MCS®-51 Family of Microcontrollers

Device Name	ROMless Version	EPROM Version	ROM Bytes	RAM Bytes	16-Bit Timers	Ckt Type
8051	8031	(8751)	4K	128	2	HMOS
8051AH	8031AH	8751H	4K	128	2	HMOS
8052AH	8032AH	8752BH	8K	256	3	HMOS
80C51BH	80C31BH	87C51	4K	128	2	CHMOS
83C51FA	80C51FA	87C51FA	8K	256	4	CHMOS
83C152	80C152	87C152	8K	256	2	CHMOS

8051

The 8051 is the original member of the MCS-51 Family, and has been in production since 1981. Among the features of the 8051 are:

- 8-bit CPU optimized for control applications
- Extensive Boolean processing (single-bit logic) capabilities
- 32 bidirectional and individually addressable I/O lines
- 128 bytes of on-chip Data RAM
- Two 16-bit timer/counters
- Full duplex UART
- 5-source interrupt structure with 2 priority levels
- On-chip clock oscillator
- 4K bytes of on-chip Program Memory
- 64K Program Memory address space
- 64K Data Memory address space

The 8031 differs from the 8051 in not having the on-chip Program ROM. Instead, the 8031 fetches all instructions from external memory.

The EPROM version of the 8051, the 8751, is no longer in production. It has been superseded by the 8751H.

8051AH

The 8051AH is identical to the 8051, but is fabricated with HMOS II technology. It is pin-for-pin compatible with the 8051.

The ROMless version of the 8051AH is the 8031AH. The EPROM version is the 8751H.

8052AH

The 8052AH is an enhanced 8051. It is fabricated with HMOS II technology, and is backwards compatible with the 8051. Its enhancements over the 8051 are as follows:

- 256 bytes of on-chip Data RAM
- Three timer/counters
- 6-source interrupt structure
- 8K bytes of on-chip Program ROM

The ROMless version of the 8052AH is the 8032AH. The EPROM version is the 8752BH.

A separate product, the 8052AH-BASIC, is an 8052AH with a full BASIC interpreter in the on-chip ROM.

80C51BH

The 80C51BH is the CHMOS version of the 8051. Functionally, it is fully compatible with the 8051, but being CMOS it draws less current than its HMOS counterpart. To further exploit the power savings available in CMOS circuitry, two reduced power modes are added:

- Software-invoked Idle Mode, during which the CPU is turned off while the RAM and other on-chip peripherals continue operating. In this mode, current draw is reduced to about 15% of the current drawn when the device is fully active.
- Software-invoked Power Down Mode, during which all on-chip activities are suspended. The on-chip RAM continues to hold its data. In this mode the device typically draws less than 10 μA.

Although the 80C51BH is functionally compatible with its HMOS counterpart, specific differences between the two types of devices must be considered in the design of an application circuit if one wishes to ensure complete interchangeability between the HMOS and CHMOS devices. These considerations are discussed in the Application Note AP-252, "Designing with the 80C51BH".

The ROMless version of the 80C51BH is the 80C31BH. The EPROM version is the 87C51.

83C51FA

The 83C51FA is an enhanced version of the 80C51BH and is backwards compatible with the 80C51BH. The new features which have been incorporated are as follows:

- Programmable Counter Array with
  Compare/Capture
  High Speed Output
  Pulse Width Modulator
  Watchdog Timer
- Programmable Serial Channel
  Automatic Address Recognition
  Framing Error Detection
- Enhanced Power Down Mode
- Up/down timer/counter
- 8 Kbytes of on-chip Program ROM
- 256 bytes of on-chip Data RAM
- 7-source interrupt structure

For further information on these new features, refer to the "Hardware Description of the 83C51FA" chapter.

The ROMless version of the 83C51FA is the 80C51FA. The EPROM version is the 87C51FA.

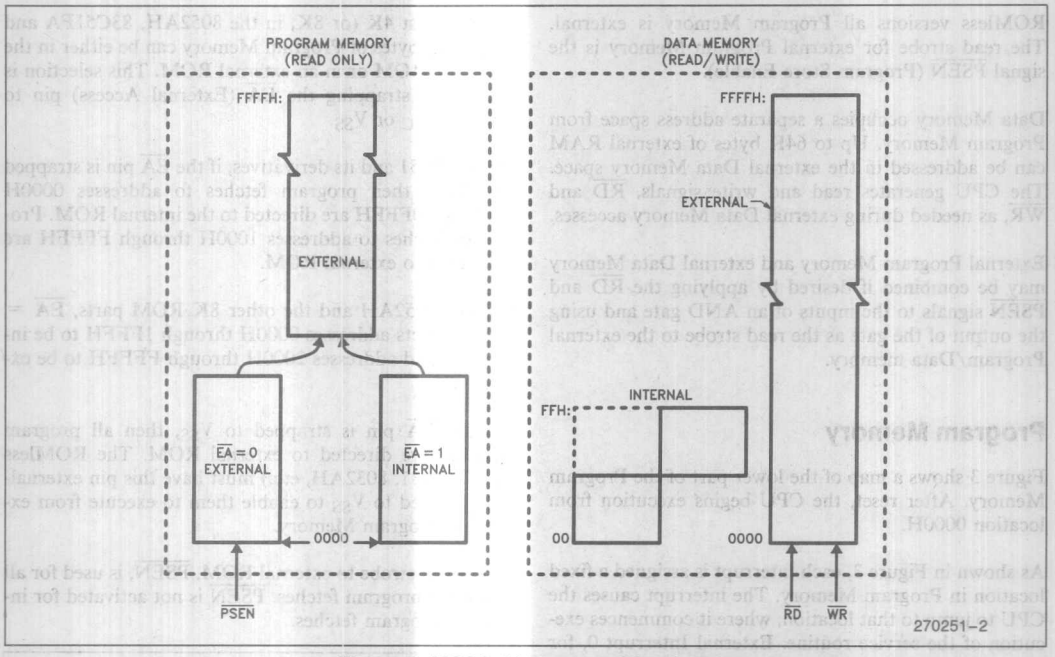


Figure 2. MCS®-51 Memory Structure

83C152

The 83C152 is an enhanced version of the 80C51BH and is 100% compatible with code written for the 80C51BH. Some of the new features which have been incorporated are:

- Global Serial Channel (GSC)—
  A high speed serial communication link capable of transmitting data in excess of 2 Mbps in either HDLC or CSMA/CD protocols.
- Two DMA Channels—
  Each DMA channel is capable of transferring 64 Kbytes of data. Options include: automatic addressing, automatic servicing of the GSC or UART, alternate cycle transfers, and burst data transfers. The source and/or destination can be internal RAM, external RAM, or SFR memory space. Most DMA transfers take 1 machine cycle to complete.
- Port 4—
  The 83C152 has added an additional port called port 4. Because of the added port, the 83C152 is available in 48-pin DIP or 68-pin PLCC packages.
- 8 Kbytes of on-chip program ROM
- 256 bytes of on-chip data RAM
- 11 interrupt vectors

For more information on the 83C152 please refer to the "Hardware Description" chapter on this product.

The ROMless version of the 83C152 is the 80C152. There is no EPROM version for the 83C152 but a version is offered which has an additional two ports which can be connected to an EPROM for ROM development.

MEMORY ORGANIZATION IN MCS®-51 DEVICES

Logical Separation of Program and Data Memory

All MCS-51 devices have separate address spaces for Program and Data Memory, as shown in Figure 2. The logical separation of Program and Data Memory allows the Data Memory to be accessed by 8-bit addresses, which can be more quickly stored and manipulated by an 8-bit CPU. Nevertheless, 16-bit Data Memory addresses can also be generated through the DPTR register.

Program Memory can only be read, not written to. There can be up to 64K bytes of Program Memory. In the 8051, 8051AH, 80C51BH, and their EPROM versions, the lowest 4K bytes of Program Memory are on-chip. The 8052AH, 83C51FA, and 83C152 provide 8 Kbytes of on-chip Program Memory storage. In the

ROMless versions all Program Memory is external. The read strobe for external Program Memory is the signal $\overline{PSEN}$ (Program Store Enable).

Data Memory occupies a separate address space from Program Memory. Up to 64K bytes of external RAM can be addressed in the external Data Memory space. The CPU generates read and write signals, $\overline{RD}$ and $\overline{WR}$, as needed during external Data Memory accesses.

External Program Memory and external Data Memory may be combined if desired by applying the $\overline{RD}$ and $\overline{PSEN}$ signals to the inputs of an AND gate and using the output of the gate as the read strobe to the external Program/Data memory.

Program Memory

Figure 3 shows a map of the lower part of the Program Memory. After reset, the CPU begins execution from location 0000H.

As shown in Figure 3, each interrupt is assigned a fixed location in Program Memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose Program Memory.

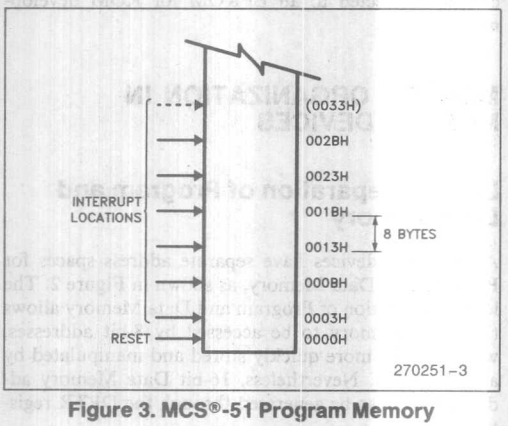


Figure 3. MCS®-51 Program Memory

The interrupt service locations are spaced at 8-byte intervals: 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1, 001BH for Timer 1, etc. If an interrupt service routine is short enough (as is often the case in control applications), it can reside entirely within that 8-byte interval. Longer service routines can use a jump instruction to skip over subsequent interrupt locations, if other interrupts are in use.

The lowest 4K (or 8K, in the 8052AH, 83C51FA and 83C152) bytes of Program Memory can be either in the on-chip ROM or in an external ROM. This selection is made by strapping the $\overline{EA}$ (External Access) pin to either V_{CC} or V_{SS}.

In the 8051 and its derivatives, if the $\overline{EA}$ pin is strapped to V_{CC}, then program fetches to addresses 0000H through 0FFFH are directed to the internal ROM. Program fetches to addresses 1000H through FFFFH are directed to external ROM.

In the 8052AH and the other 8K ROM parts, $\overline{EA}$ = V_{CC} selects addresses 0000H through 1FFFH to be internal, and addresses 2000H through FFFFH to be external.

If the $\overline{EA}$ pin is strapped to V_{SS}, then all program fetches are directed to external ROM. The ROMless parts (8031, 8032AH, etc.) must have this pin externally strapped to V_{SS} to enable them to execute from external Program Memory.

The read strobe to external ROM, $\overline{PSEN}$, is used for all external program fetches. $\overline{PSEN}$ is not activated for internal program fetches.

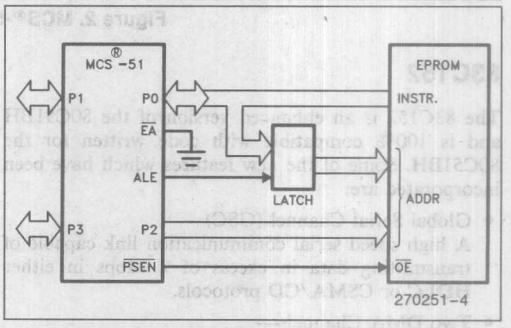


Figure 4. Executing from External Program Memory

The hardware configuration for external program execution is shown in Figure 4. Note that 16 I/O lines (Ports 0 and 2) are dedicated to bus functions during external Program Memory fetches. Port 0 (P0 in Figure 4) serves as a multiplexed address/data bus. It emits the low byte of the Program Counter (PCL) as an address, and then goes into a float state awaiting the arrival of the code byte from the Program Memory. During the time that the low byte of the Program Counter is valid on P0, the signal ALE (Address Latch Enable) clocks this byte into an address latch. Meanwhile, Port 2 (P2 in Figure 4) emits the high byte of the Program Counter (PCH). Then $\overline{PSEN}$ strobes the EPROM and the code byte is read into the microcontroller.

Program Memory addresses are always 16 bits wide, even though the actual amount of Program Memory used may be less than 64K bytes. External program execution sacrifices two of the 8-bit ports, P0 and P2, to the function of addressing the Program Memory.

Data Memory

The right half of Figure 2 shows the internal and external Data Memory spaces available to the MCS-51 user.

Figure 5 shows a hardware configuration for accessing up to 2K bytes of external RAM. The CPU in this case is executing from internal ROM. Port 0 serves as a multiplexed address/data bus to the RAM, and 3 lines of Port 2 are being used to page the RAM. The CPU generates $\overline{RD}$ and $\overline{WR}$ signals as needed during external RAM accesses.

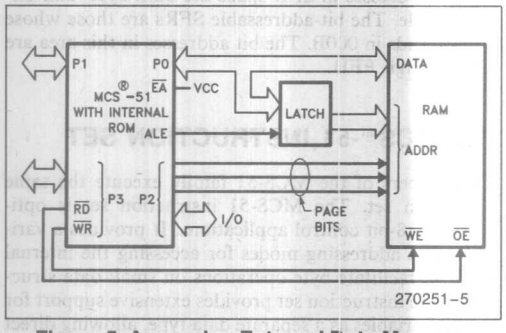


Figure 5. Accessing External Data Memory. If the Program Memory is Internal, the Other Bits of P2 are Available as I/O.

There can be up to 64K bytes of external Data Memory. External Data Memory addresses can be either 1 or 2 bytes wide. One-byte addresses are often used in conjunction with one or more other I/O lines to page the RAM, as shown in Figure 5. Two-byte addresses can also be used, in which case the high address byte is emitted at Port 2.

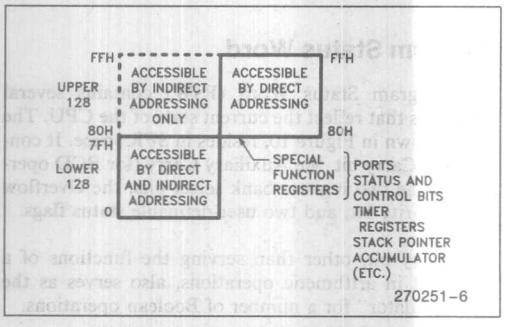


Figure 6. Internal Data Memory

Internal Data Memory is mapped in Figure 6. The memory space is shown divided into three blocks, which are generally referred to as the Lower 128, the Upper 128, and SFR space.

Internal Data Memory addresses are always one byte wide, which implies an address space of only 256 bytes. However, the addressing modes for internal RAM can in fact accommodate 384 bytes, using a simple trick. Direct addresses higher than 7FH access one memory space, and indirect addresses higher than 7FH access a different memory space. Thus Figure 6 shows the Upper 128 and SFR space occupying the same block of addresses, 80H through FFH, although they are physically separate entities.

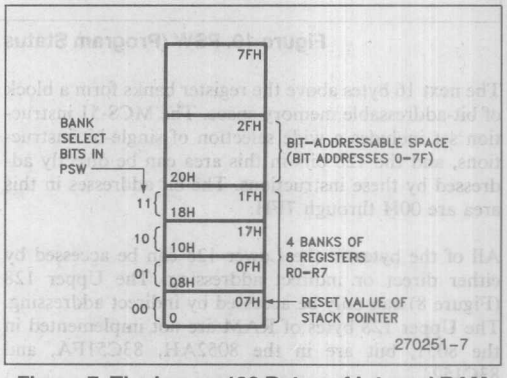


Figure 7. The Lower 128 Bytes of Internal RAM

The Lower 128 bytes of RAM are present in all MCS-51 devices as mapped in Figure 7. The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing.

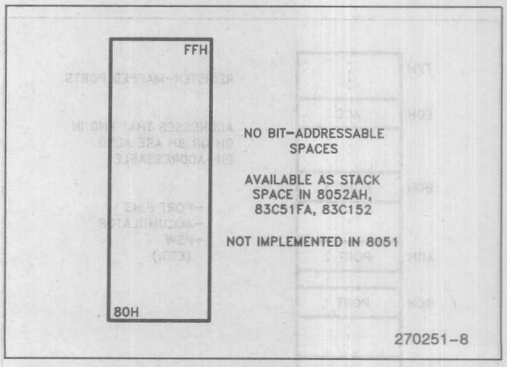


Figure 8. The Upper 128 Bytes of Internal RAM

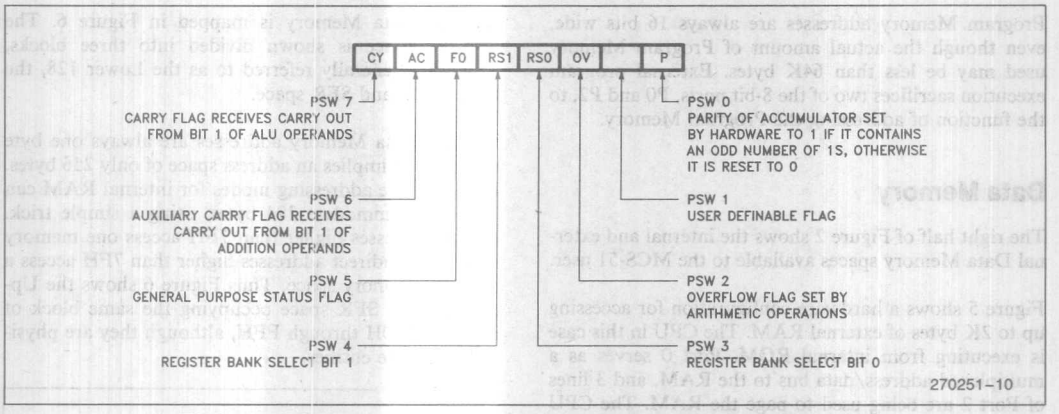


Figure 10. PSW (Program Status Word) Register in MCS®-51 Devices

The next 16 bytes above the register banks form a block of bit-addressable memory space. The MCS-51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00H through 7FH.

All of the bytes in the Lower 128 can be accessed by either direct or indirect addressing. The Upper 128 (Figure 8) can only be accessed by indirect addressing. The Upper 128 bytes of RAM are not implemented in the 8051, but are in the 8052AH, 83C51FA, and 83C152.

Figure 9 gives a brief look at the Special Function Register (SFR) space. SFRs include the Port latches, timers, peripheral controls, etc. These registers can only be accessed by direct addressing. In general, all MCS-51 microcontrollers have the same SFRs as the 8051, and at the same addresses in SFR space. However, enhancements to the 8051 have additional SFRs that are not present in the 8051, nor perhaps in other proliferations of the family.

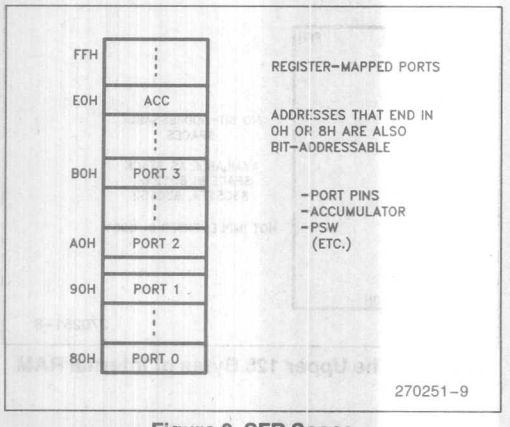


Figure 9. SFR Space

Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 000B. The bit addresses in this area are 80H through FFH.

THE MCS®-51 INSTRUCTION SET

All members of the MCS-51 family execute the same instruction set. The MCS-51 instruction set is optimized for 8-bit control applications. It provides a variety of fast addressing modes for accessing the internal RAM to facilitate byte operations on small data structures. The instruction set provides extensive support for one-bit variables as a separate data type, allowing direct bit manipulation in control and logic systems that require Boolean processing.

An overview of the MCS-51 instruction set is presented below, with a brief description of how certain instructions might be used. References to "the assembler" in this discussion are to Intel's MCS-51 Macro Assembler, ASM51. More detailed information on the instruction set can be found in the MCS-51 Macro Assembler User's Guide (Order No. 9800937 for ISIS Systems, Order No. 122752 for DOS Systems).

Program Status Word

The Program Status Word (PSW) contains several status bits that reflect the current state of the CPU. The PSW, shown in Figure 10, resides in SFR space. It contains the Carry bit, the Auxiliary Carry (for BCD operations), the two register bank select bits, the Overflow flag, a Parity bit, and two user-definable status flags.

The Carry bit, other than serving the functions of a Carry bit in arithmetic operations, also serves as the "Accumulator" for a number of Boolean operations.

The bits RS0 and RS1 are used to select one of the four register banks shown in Figure 7. A number of instructions refer to these RAM locations as R0 through R7. The selection of which of the four banks is being referred to is made on the basis of the bits RS0 and RS1 at execution time.

The Parity bit reflects the number of 1s in the Accumulator: P = 1 if the Accumulator contains an odd number of 1s, and P = 0 if the Accumulator contains an even number of 1s. Thus the number of 1s in the Accumulator plus P is always even.

Two bits in the PSW are uncommitted and may be used as general purpose status flags.

Addressing Modes

The addressing modes in the MCS-51 instruction set are as follows:

DIRECT ADDRESSING

In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal Data RAM and SFRs can be directly addressed.

INDIRECT ADDRESSING

In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed.

The address register for 8-bit addresses can be R0 or R1 of the selected register bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit "data pointer" register, DPTR.

REGISTER INSTRUCTIONS

The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed. One of four banks is selected at execution time by the two bank select bits in the PSW.

REGISTER-SPECIFIC INSTRUCTIONS

Some instructions are specific to a certain register. For example, some instructions always operate on the Accumulator, or Data Pointer, etc., so no address byte is needed to point to it. The opcode itself does that. Instructions that refer to the Accumulator as A assemble as accumulator-specific opcodes.

IMMEDIATE CONSTANTS

The value of a constant can follow the opcode in Program Memory. For example,

```
MOV A, #100
```

loads the Accumulator with the decimal number 100. The same number could be specified in hex digits as 64H.

INDEXED ADDRESSING

Only Program Memory can be accessed with indexed addressing, and it can only be read. This addressing mode is intended for reading look-up tables in Program Memory. A 16-bit base register (either DPTR or the Program Counter) points to the base of the table, and the Accumulator is set up with the table entry number. The address of the table entry in Program Memory is formed by adding the Accumulator data to the base pointer.

Another type of indexed addressing is used in the "case jump" instruction. In this case the destination address of a jump instruction is computed as the sum of the base pointer and the Accumulator data.

Arithmetic Instructions

The menu of arithmetic instructions is listed in Table 2. The table indicates the addressing modes that can be used with each instruction to access the <byte> operand. For example, the ADD A,<byte> instruction can be written as:

```
ADD     A,7FH      (direct addressing)
ADD     A,@R0      (indirect addressing)
ADD     A,R7       (register addressing)
ADD     A,#127     (immediate constant)
```

The execution times listed in Table 2 assume a 12 MHz clock frequency. All of the arithmetic instructions execute in 1 μs except the INC DPTR instruction, which takes 2 μs, and the Multiply and Divide instructions, which take 4 μs.

Note that any byte in the internal Data Memory space can be incremented or decremented without going through the Accumulator.

One of the INC instructions operates on the 16-bit Data Pointer. The Data Pointer is used to generate 16-bit addresses for external memory, so being able to increment it in one 16-bit operation is a useful feature.

The MUL AB instruction multiplies the Accumulator by the data in the B register and puts the 16-bit product into the concatenated B and Accumulator registers.

Table 2. A List of the MCS®-51 Arithmetic Instructions

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
ADD A,<byte>	A = A + <byte>	X	X	X	X	1
ADDC A,<byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A,<byte>	A = A − <byte> − C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A − 1	Accumulator only				1
DEC <byte>	<byte> = <byte> − 1	X	X	X		1
MUL AB	B:A = B x A	ACC and B only				4
DIV AB	A = Int [A/B] B = Mod [A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

The DIV AB instruction divides the Accumulator by the data in the B register and leaves the 8-bit quotient in the Accumulator, and the 8-bit remainder in the B register.

Oddly enough, DIV AB finds less use in arithmetic "divide" routines than in radix conversions and programmable shift operations. An example of the use of DIV AB in a radix conversion will be given later. In shift operations, dividing a number by 2^n shifts its n bits to the right. Using DIV AB to perform the division completes the shift in 4 μs and leaves the B register holding the bits that were shifted out.

The DA A instruction is for BCD arithmetic operations. In BCD arithmetic, ADD and ADDC instructions should always be followed by a DA A operation, to ensure that the result is also in BCD. Note that DA A will not convert a binary number to BCD. The DA A operation produces a meaningful result only as the second step in the addition of two BCD bytes.

Table 3. A List of the MCS®-51 Logical Instructions

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
ANL A,<byte>	A = A .AND. <byte>	X	X	X	X	1
ANL <byte>,A	<byte> = <byte> .AND. A	X				1
ANL <byte>,#data	<byte> = <byte> .AND. #data	X				2
ORL A,<byte>	A = A .OR. <byte>	X	X	X	X	1
ORL <byte>,A	<byte> = <byte> .OR. A	X				1
ORL <byte>,#data	<byte> = <byte> .OR. #data	X				2
XRL A,<byte>	A = A .XOR. <byte>	X	X	X	X	1
XRL <byte>,A	<byte> = <byte> .XOR. A	X				1
XRL <byte>,#data	<byte> = <byte> .XOR. #data	X				2
CRL A	A = 00H	Accumulator only				1
CPL A	A = .NOT. A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

Logical Instructions

Table 3 shows the list of MCS-51 logical instructions. The instructions that perform Boolean operations (AND, OR, Exclusive OR, NOT) on bytes perform the operation on a bit-by-bit basis. That is, if the Accumulator contains 00110101B and <byte> contains 01010011B, then

```
ANL    A,<byte>
```

will leave the Accumulator holding 00010001B.

The addressing modes that can be used to access the <byte> operand are listed in Table 3. Thus, the ANL A,<byte> instruction may take any of the forms

```
ANL    A,7FH      (direct addressing)
ANL    A,@R1      (indirect addressing)
ANL    A,R6       (register addressing)
ANL    A,#53H     (immediate constant)
```

All of the logical instructions that are Accumulator-specific execute in 1μs (using a 12 MHz clock). The others take 2 μs.

Note that Boolean operations can be performed on any byte in the internal Data Memory space without going through the Accumulator. The XRL <byte>, #data instruction, for example, offers a quick and easy way to invert port bits, as in

```
XRL    P1,#0FFH
```

If the operation is in response to an interrupt, not using the Accumulator saves the time and effort to stack it in the service routine.

The Rotate instructions (RL A, RLC A, etc.) shift the Accumulator 1 bit to the left or right. For a left rotation, the MSB rolls into the LSB position. For a right rotation, the LSB rolls into the MSB position.

The SWAP A instruction interchanges the high and low nibbles within the Accumulator. This is a useful operation in BCD manipulations. For example, if the Accumulator contains a binary number which is known to be less than 100, it can be quickly converted to BCD by the following code:

```
MOV    B,#10
DIV    AB
SWAP   A
ADD    A,B
```

Dividing the number by 10 leaves the tens digit in the low nibble of the Accumulator, and the ones digit in the B register. The SWAP and ADD instructions move the tens digit to the high nibble of the Accumulator, and the ones digit to the low nibble.

Data Transfers

INTERNAL RAM

Table 4 shows the menu of instructions that are available for moving data around within the internal memory spaces, and the addressing modes that can be used with each one. With a 12 MHz clock, all of these instructions execute in either 1 or 2 μs.

The MOV <dest>, <src> instruction allows data to be transferred between any two internal RAM or SFR locations without going through the Accumulator. Remember the Upper 128 byes of data RAM can be accessed only by indirect addressing, and SFR space only by direct addressing.

Note that in all MCS-51 devices, the stack resides in on-chip RAM, and grows upwards. The PUSH instruction first increments the Stack Pointer (SP), then copies the byte into the stack. PUSH and POP use only direct addressing to identify the byte being saved or restored,

Table 4. A List of the MCS®-51 Data Transfer Instructions that Access Internal Data Memory Space

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
MOV A,<src>	A = <src>	X	X	X	X	1
MOV <dest>,A	<dest> = A	X	X	X		1
MOV <dest>, <src>	<dest> = <src>	X	X	X	X	2
MOV DPTR,#data16	DPTR = 16-bit immediate constant.				X	2
PUSH <src>	INC SP : MOV "@SP",<src>	X				2
POP <dest>	MOV <dest>, "@SP" : DEC SP	X				2
XCH A,<byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A,@Ri	ACC and @Ri exchange low nibbles		X			1

but the stack itself is accessed by indirect addressing using the SP register. This means the stack can go into the Upper 128, if they are implemented, but not into SFR space.

The Upper 128 are not implemented in the 8051, 8051AH, or 80C51BH, nor in their ROMless or EPROM counterparts. With these devices, if the SP points to the Upper 128, PUSHed bytes are lost, and POPped bytes are indeterminate.

The Data Transfer instructions include a 16-bit MOV that can be used to initialize the Data Pointer (DPTR) for look-up tables in Program Memory, or for 16-bit external Data Memory accesses.

The XCH A, <byte> instruction causes the Accumulator and addressed byte to exchange data. The XCHD A,@Ri instruction is similar, but only the low nibbles are involved in the exchange.

To see how XCH and XCHD can be used to facilitate data manipulations, consider first the problem of shifting an 8-digit BCD number two digits to the right. Figure 11 shows how this can be done using direct MOVs, and for comparison how it can be done using XCH instructions. To aid in understanding how the code works, the contents of the registers that are holding the BCD number and the content of the Accumulator are shown alongside each instruction to indicate their status after the instruction has been executed.

		2A	2B	2C	2D	2E	ACC
MOV	A,2EH	00	12	34	56	78	78
MOV	2EH,2DH	00	12	34	56	56	78
MOV	2DH,2CH	00	12	34	34	56	78
MOV	2CH,2BH	00	12	12	34	56	78
MOV	2BH,#0	00	00	12	34	56	78

(a) Using direct MOVs: 14 bytes, 9 μs

		2A	2B	2C	2D	2E	ACC
CLR	A	00	12	34	56	78	00
XCH	A,2BH	00	00	34	56	78	12
XCH	A,2CH	00	00	12	56	78	34
XCH	A,2DH	00	00	12	34	78	56
XCH	A,2EH	00	00	12	34	56	78

(b) Using XCHs: 9 bytes, 5 μs

Figure 11. Shifting a BCD Number Two Digits to the Right

After the routine has been executed, the Accumulator contains the two digits that were shifted out on the right. Doing the routine with direct MOVs uses 14 code bytes and 9 μs of execution time (assuming a 12 MHz clock). The same operation with XCHs uses less code and executes almost twice as fast.

To right-shift by an odd number of digits, a one-digit shift must be executed. Figure 12 shows a sample of code that will right-shift a BCD number one digit, using the XCHD instruction. Again, the contents of the registers holding the number and of the Accumulator are shown alongside each instruction.

		2A	2B	2C	2D	2E	ACC
	MOV R1,#2EH	00	12	34	56	78	XX
	MOV R0,#2DH	00	12	34	56	78	XX
	loop for R1 = 2EH:						
LOOP:	MOV A,@R1	00	12	34	56	78	78
	XCHD A,@R0	00	12	34	58	78	76
	SWAP A	00	12	34	58	78	67
	MOV @R1,A	00	12	34	58	67	67
	DEC R1	00	12	34	58	67	67
	DEC R0	00	12	34	58	67	67
	CJNE R1,#2AH,LOOP						
	loop for R1 = 2DH:	00	12	38	45	67	45
	loop for R1 = 2CH:	00	18	23	45	67	23
	loop for R1 = 2BH:	08	01	23	45	67	01
	CLR A	08	01	23	45	67	00
	XCH A,2AH	00	01	23	45	67	08

Figure 12. Shifting a BCD Number One Digit to the Right

First, pointers R1 and R0 are set up to point to the two bytes containing the last four BCD digits. Then a loop is executed which leaves the last byte, location 2EH, holding the last two digits of the shifted number. The pointers are decremented, and the loop is repeated for location 2DH. The CJNE instruction (Compare and Jump if Not Equal) is a loop control that will be described later.

The loop is executed from LOOP to CJNE for R1 = 2EH, 2DH, 2CH and 2BH. At that point the digit that was originally shifted out on the right has propagated to location 2AH. Since that location should be left with 0s, the lost digit is moved to the Accumulator.

EXTERNAL RAM

Table 5 shows a list of the Data Transfer instructions that access external Data Memory. Only indirect addressing can be used. The choice is whether to use a one-byte address, @Ri, where Ri can be either R0 or R1 of the selected register bank, or a two-byte address, @DPTR. The disadvantage to using 16-bit addresses if only a few K bytes of external RAM are involved is that 16-bit addresses use all 8 bits of Port 2 as address bus. On the other hand, 8-bit addresses allow one to address a few K bytes of RAM, as shown in Figure 5, without having to sacrifice all of Port 2.

All of these instructions execute in 2 μs, with a 12 MHz clock.

Table 5. A List of the MCS®-51 Data Transfer Instructions that Access External Data Memory Space

Address Width	Mnemonic	Operation	Execution Time (μs)
8 bits	MOVX A,@Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri,A	Write external RAM @Ri	2
16 bits	MOVX A,@DPTR	Read external RAM @DPTR	2
16 bits	MOVX @DPTR,A	Write external RAM @DPTR	2

Note that in all external Data RAM accesses, the Accumulator is always either the destination or source of the data.

The read and write strobes to external RAM are activated only during the execution of a MOVX instruction. Normally these signals are inactive, and in fact if they're not going to be used at all, their pins are available as extra I/O lines. More about that later.

LOOKUP TABLES

Table 6 shows the two instructions that are available for reading lookup tables in Program Memory. Since these instructions access only Program Memory, the lookup tables can only be read, not updated. The mnemonic is MOVC for "move constant".

If the table access is to external Program Memory, then the read strobe is $\overline{PSEN}$.

Table 6. The MCS®-51 Lookup Table Read Instructions

Mnemonic	Operation	Execution Time (μs)
MOVC A,@A+DPTR	Read Pgm Memory at (A+DPTR)	2
MOVC A,@A+PC	Read Pgm Memory at (A+PC)	2

The first MOVC instruction in Table 6 can accommodate a table of up to 256 entries, numbered 0 through 255. The number of the desired entry is loaded into the Accumulator, and the Data Pointer is set up to point to beginning of the table. Then

```
MOVC    A,@A+DPTR
```

copies the desired table entry into the Accumulator.

The other MOVC instruction works the same way, except the Program Counter (PC) is used as the table base, and the table is accessed through a subroutine. First the number of the desired entry is loaded into the Accumulator, and the subroutine is called:

```
        MOV     A,ENTRY_NUMBER
        CALL    TABLE
```

The subroutine "TABLE" would look like this:

```
TABLE:  MOVC    A,@A+PC
        RET
```

The table itself immediately follows the RET (return) instruction in Program Memory. This type of table can have up to 255 entries, numbered 1 through 255. Number 0 can not be used, because at the time the MOVC instruction is executed, the PC contains the address of the RET instruction. An entry numbered 0 would be the RET opcode itself.

Boolean Instructions

MCS-51 devices contain a complete Boolean (single-bit) processor. The internal RAM contains 128 addressable bits, and the SFR space can support up to 128 other addressable bits. All of the port lines are bit-addressable, and each one can be treated as a separate single-bit port. The instructions that access these bits are not just conditional branches, but a complete menu of move, set, clear, complement, OR, and AND instructions. These kinds of bit operations are not easily obtained in other architectures with any amount of byte-oriented software.

Table 7. A List of the MCS®-51 Boolean Instructions

Mnemonic		Operation	Execution Time (μs)
ANL	C,bit	C = C .AND. bit	2
ANL	C,/bit	C = C .AND. .NOT. bit	2
ORL	C,bit	C = C .OR. bit	2
ORL	C,/bit	C = C .OR. .NOT. bit	2
MOV	C,bit	C = bit	1
MOV	bit,C	bit = C	2
CLR	C	C = 0	1
CLR	bit	bit = 0	1
SETB	C	C = 1	1
SETB	bit	bit = 1	1
CPL	C	C = .NOT. C	1
CPL	bit	bit = .NOT. bit	1
JC	rel	Jump if C = 1	2
JNC	rel	Jump if C = 0	2
JB	bit,rel	Jump if bit = 1	2
JNB	bit,rel	Jump if bit = 0	2
JBC	bit,rel	Jump if bit = 1; CLR bit	2

The instruction set for the Boolean processor is shown in Table 7. All bit accesses are by direct addressing. Bit addresses 00H through 7FH are in the Lower 128, and bit addresses 80H through FFH are in SFR space.

Note how easily an internal flag can be moved to a port pin:

```
MOV   C,FLAG
MOV   P1.0,C
```

In this example, FLAG is the name of any addressable bit in the Lower 128 or SFR space. An I/O line (the LSB of Port 1, in this case) is set or cleared depending on whether the flag bit is 1 or 0.

The Carry bit in the PSW is used as the single-bit Accumulator of the Boolean processor. Bit instructions that refer to the Carry bit as C assemble as Carry-specific instructions (CLR C, etc). The Carry bit also has a direct address, since it resides in the PSW register, which is bit-addressable.

Note that the Boolean instruction set includes ANL and ORL operations, but not the XRL (Exclusive OR) operation. An XRL operation is simple to implement in software. Suppose, for example, it is required to form the Exclusive OR of two bits:

C = bit1 .XRL. bit2

The software to do that could be as follows:

```
      MOV    C,bit1
      JNB    bit2,OVER
      CPL    C
OVER: (continue)
```

First, bit1 is moved to the Carry. If bit2 = 0, then C now contains the correct result. That is, bit1 .XRL. bit2 = bit1 if bit2 = 0. On the other hand, if bit2 = 1 C now contains the complement of the correct result. It need only be inverted (CPL C) to complete the operation.

This code uses the JNB instruction, one of a series of bit-test instructions which execute a jump if the addressed bit is set (JC, JB, JBC) or if the addressed bit is not set (JNC, JNB). In the above case, bit2 is being tested, and if bit2 = 0 the CPL C instruction is jumped over.

JBC executes the jump if the addressed bit is set, and also clears the bit. Thus a flag can be tested and cleared in one operation.

All the PSW bits are directly addressable, so the Parity bit, or the general purpose flags, for example, are also available to the bit-test instructions.

RELATIVE OFFSET

The destination address for these jumps is specified to the assembler by a label or by an actual address in Program Memory. However, the destination address assembles to a **relative** offset byte. This is a signed (two's complement) offset byte which is added to the PC in two's complement arithmetic if the jump is executed.

The range of the jump is therefore −128 to +127 Program Memory bytes relative to the first byte following the instruction.

Jump Instructions

Table 8 shows the list of unconditional jumps.

Table 8. Unconditional Jumps in MCS®-51 Devices

Mnemonic	Operation	Execution Time (μs)
JMP addr	Jump to addr	2
JMP @A+DPTR	Jump to A+DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

The Table lists a single "JMP addr" instruction, but in fact there are three—SJMP, LJMP and AJMP—which differ in the format of the destination address. JMP is a generic mnemonic which can be used if the programmer does not care which way the jump is encoded.

The SJMP instruction encodes the destination address as a relative offset, as described above. The instruction is 2 bytes long, consisting of the opcode and the relative offset byte. The jump distance is limited to a range of −128 to +127 bytes relative to the instruction following the SJMP.

The LJMP instruction encodes the destination address as a 16-bit constant. The instruction is 3 bytes long, consisting of the opcode and two address bytes. The destination address can be anywhere in the 64K Program Memory space.

The AJMP instruction encodes the destination address as an 11-bit constant. The instruction is 2 bytes long, consisting of the opcode, which itself contains 3 of the 11 address bits, followed by another byte containing the low 8 bits of the destination address. When the instruction is executed, these 11 bits are simply substituted for the low 11 bits in the PC. The high 5 bits stay the same. Hence the destination has to be within the same 2K block as the instruction following the AJMP.

In all cases the programmer specifies the destination address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the destination address into the correct format for the given instruction. If the format required by the instruction will not support the distance to the specified destination address, a "Destination out of range" message is written into the List file.

The JMP @A+DPTR instruction supports case jumps. The destination address is computed at execution time as the sum of the 16-bit DPTR register and the Accumulator. Typically, DPTR is set up with the address of a jump table, and the Accumulator is given an index to the table. In a 5-way branch, for example, an integer 0 through 4 is loaded into the Accumulator. The code to be executed might be as follows:

```
MOV   DPTR,#JUMP_TABLE
MOV   A,INDEX_NUMBER
RL    A
JMP   @A+DPTR
```

The RL A instruction converts the index number (0 through 4) to an even number on the range 0 through 8, because each entry in the jump table is 2 bytes long:

```
JUMP_TABLE:
      AJMP    CASE_0
      AJMP    CASE_1
      AJMP    CASE_2
      AJMP    CASE_3
      AJMP    CASE_4
```

Table 8 shows a single "CALL addr" instruction, but there are two of them—LCALL and ACALL—which differ in the format in which the subroutine address is given to the CPU. CALL is a generic mnemonic which can be used if the programmer does not care which way the address is encoded.

The LCALL instruction uses the 16-bit address format, and the subroutine can be anywhere in the 64K Program Memory space. The ACALL instruction uses the 11-bit format, and the subroutine must be in the same 2K block as the instruction following the ACALL.

In any case the programmer specifies the subroutine address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the address into the correct format for the given instructions.

Subroutines should end with a RET instruction, which returns execution to the instruction following the CALL.

RETI is used to return from an interrupt service routine. The only difference between RET and RETI is that RETI tells the interrupt control system that the interrupt in progress is done. If there is no interrupt in progress at the time RETI is executed, then the RETI is functionally identical to RET.

Table 9 shows the list of conditional jumps available to the MCS-51 user. All of these jumps specify the destination address by the relative offset method, and so are limited to a jump distance of −128 to +127 bytes from the instruction following the conditional jump instruction. Important to note, however, the user specifies to the assembler the actual destination address the same way as the other jumps: as a label or a 16-bit constant.

Table 9. Conditional Jumps in MCS®-51 Devices

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A ≠ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNE A,<byte>,rel	Jump if A ≠ <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

There is no Zero bit in the PSW. The JZ and JNZ instructions test the Accumulator data for that condition.

The DJNZ instruction (Decrement and Jump if Not Zero) is for loop control. To execute a loop N times, load a counter byte with N and terminate the loop with a DJNZ to the beginning of the loop, as shown below for N = 10:

```
        MOV     COUNTER,#10
LOOP:   (begin loop)
          *
          *
          *
        (end loop)
        DJNZ    COUNTER,LOOP
        (continue)
```

The CJNE instruction (Compare and Jump if Not Equal) can also be used for loop control as in Figure 12. Two bytes are specified in the operand field of the instruction. The jump is executed only if the two bytes are not equal. In the example of Figure 12, the two bytes were the data in R1 and the constant 2AH. The initial data in R1 was 2EH. Every time the loop was executed, R1 was decremented, and the looping was to continue until the R1 data reached 2AH.

Another application of this instruction is in "greater than, less than" comparisons. The two bytes in the operand field are taken as unsigned integers. If the first is less than the second, then the Carry bit is set (1). If the first is greater than or equal to the second, then the Carry bit is cleared.

CPU TIMING

All MCS-51 microcontrollers have an on-chip oscillator which can be used if desired as the clock source for the CPU. To use the on-chip oscillator, connect a crystal or ceramic resonator between the XTAL1 and XTAL2 pins of the microcontroller, and capacitors to ground as shown in Figure 13.

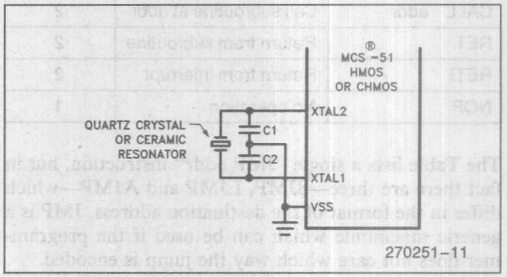
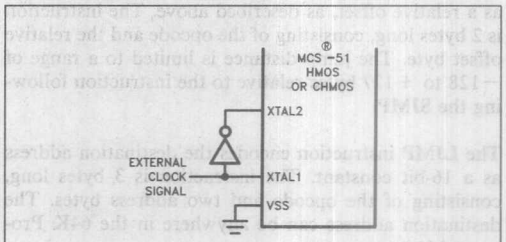
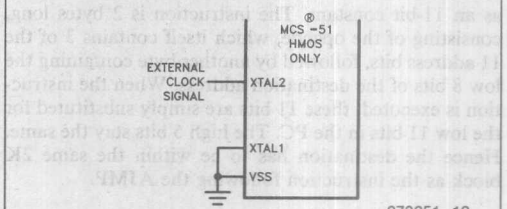


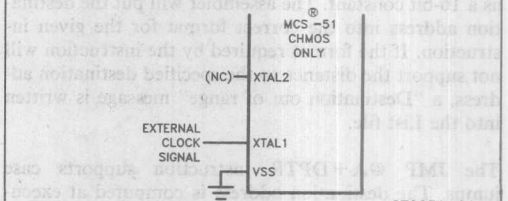
Figure 13. Using the On-Chip Oscillator



A. HMOS or CHMOS



B. HMOS Only



C. CHMOS Only

Figure 14. Using an External Clock

Examples of how to drive the clock with an external oscillator are shown in Figure 14. Note that in the HMOS devices (8051, etc.) the signal at the XTAL2 pin actually drives the internal clock generator. In the CHMOS devices (80C51BH, etc.) the signal at the XTAL1 pin drives the internal clock generator. If only one pin is going to be driven with the external oscillator signal, make sure it is the right pin.

The internal clock generator defines the sequence of states that make up the MCS-51 machine cycle.

Machine Cycles

A machine cycle consists of a sequence of 6 states, numbered S1 through S6. Each state time lasts for two oscillator periods. Thus a machine cycle takes 12 oscillator periods or 1 μs if the oscillator frequency is 12 MHz.

Each state is divided into a Phase 1 half and a Phase 2 half. Figure 15 shows the fetch/execute sequences in

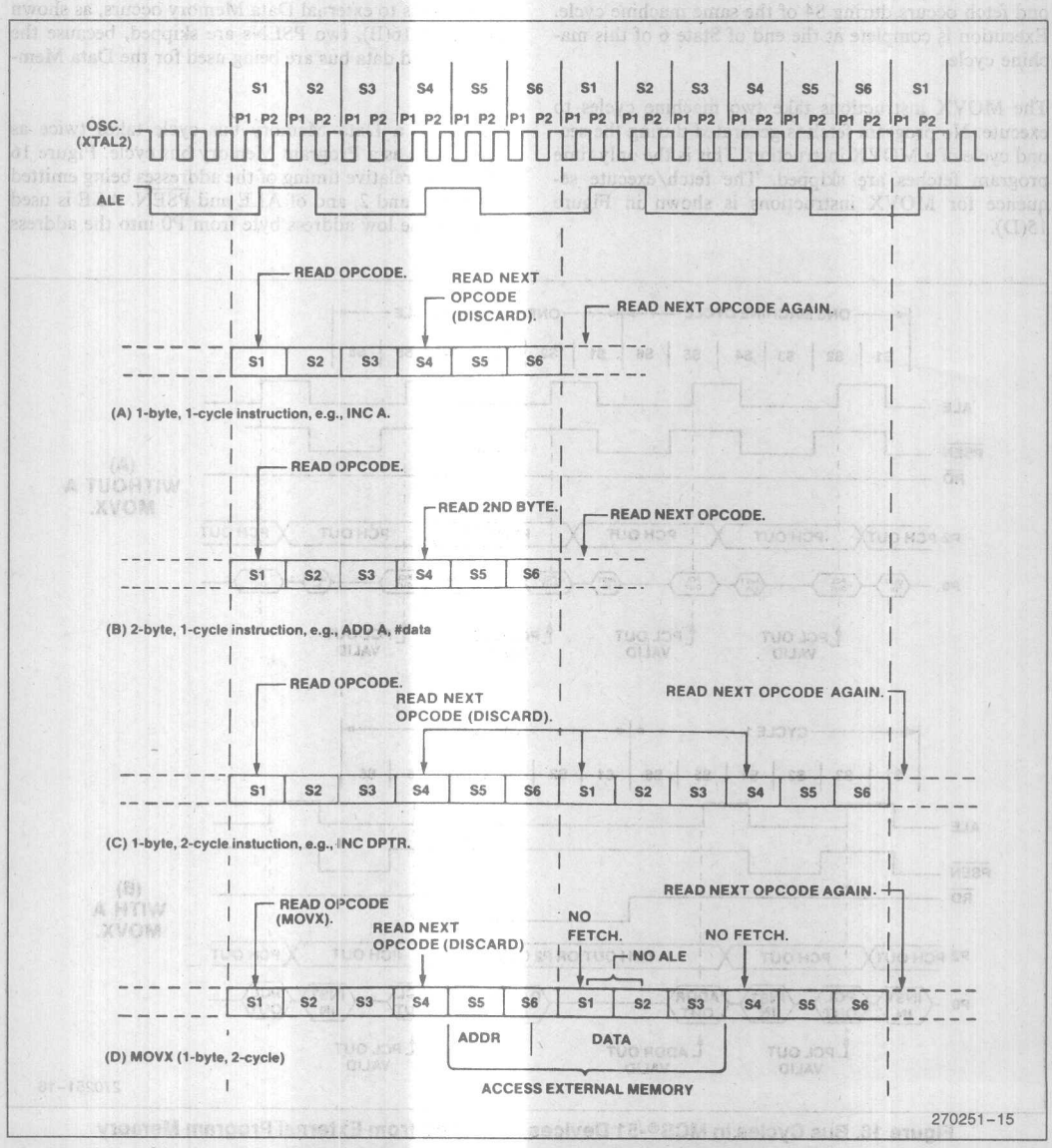


Figure 15. State Sequences in MCS®-51 Devices

states and phases for various kinds of instructions. Normally two program fetches are generated during each machine cycle, even if the instruction being executed doesn't require it. If the instruction being executed doesn't need more code bytes, the CPU simply ignores the extra fetch, and the Program Counter is not incremented.

Execution of a one-cycle instruction (Figure 15A and B) begins during State 1 of the machine cycle, when the opcode is latched into the Instruction Register. A second fetch occurs during S4 of the same machine cycle. Execution is complete at the end of State 6 of this machine cycle.

The MOVX instructions take two machine cycles to execute. No program fetch is generated during the second cycle of a MOVX instruction. This is the only time program fetches are skipped. The fetch/execute sequence for MOVX instructions is shown in Figure 15(D).

The fetch/execute sequences are the same whether the Program Memory is internal or external to the chip. Execution times do not depend on whether the Program Memory is internal or external.

Figure 16 shows the signals and timing involved in program fetches when the Program Memory is external. If Program Memory is external, then the Program Memory read strobe $\overline{PSEN}$ is normally activated twice per machine cycle, as shown in Figure 16(A).

If an access to external Data Memory occurs, as shown in Figure 16(B), two $\overline{PSEN}$s are skipped, because the address and data bus are being used for the Data Memory access.

Note that a Data Memory bus cycle takes twice as much time as a Program Memory bus cycle. Figure 16 shows the relative timing of the addresses being emitted at Ports 0 and 2, and of ALE and $\overline{PSEN}$. ALE is used to latch the low address byte from P0 into the address latch.

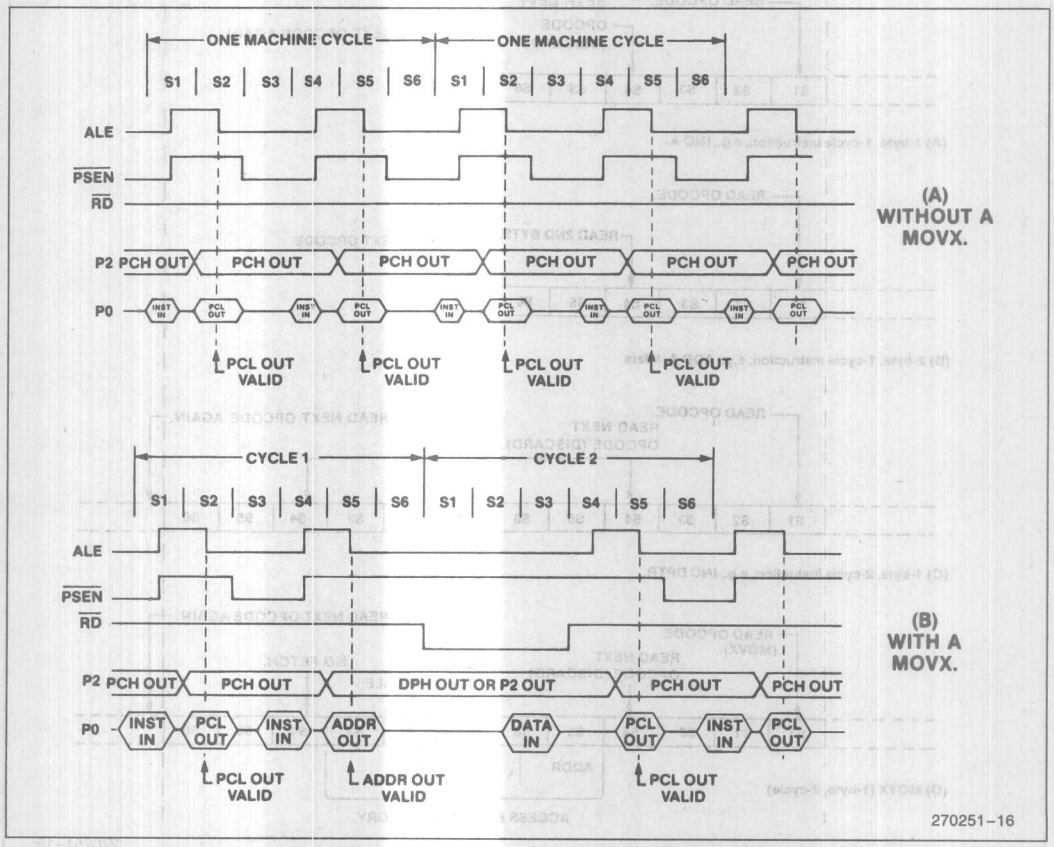


Figure 16. Bus Cycles in MCS®-51 Devices Executing from External Program Memory

When the CPU is executing from internal Program Memory, $\overline{PSEN}$ is not activated, and program addresses are not emitted. However, ALE continues to be activated twice per machine cycle and so is available as a clock output signal. Note, however, that one ALE is skipped during the execution of the MOVX instruction.

Interrupt Structure

The 8051, 8051AH, and 80C51BH, and their ROMless and EPROM versions, provide 5 interrupt sources: 2 external interrupts, 2 timer interrupts, and the serial port interrupt. The 8052AH provides these 5 plus a sixth interrupt that is associated with the third timer/counter which is present in this device. Additional interrupts are available on the 83C51FA and 83C152. Refer to the appropriate chapters on these devices for further information on their interrupts.

What follows is an overview of the interrupt structure for these devices. More detailed information for specific members of the MCS-51 family is provided in the chapters of this handbook that describe the specific devices.

INTERRUPT ENABLES

Each of the interrupt sources can be individually enabled or disabled by setting or clearing a bit in the SFR named IE (Interrupt Enable). This register also contains a global disable bit, which can be cleared to disable all interrupts at once. Figure 17 shows the IE register for the 8052AH.

(MSB) (LSB)

EA	—	ET2	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	reserved
ET2	IE.5	enables or disables the Timer 2 overflow or capture interrupt. If ET2 = 0, the Timer 2 interrupt is disabled.
ES	IE.4	enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	enables or disables External Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

Figure 17. IE (Interrupt Enable) Register in the 8052AH

INTERRUPT PRIORITIES

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in the SFR named IP (Interrupt Priority). Figure 18 shows the IP register in the 8052AH.

A low-priority interrrupt can be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

If two interrupt requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence.

Figure 19 shows, for the 8052AH, how the IE and IP registers and the polling sequence work to determine which if any interrupt will be serviced.

(MSB) (LSB)

—	—	PT2	PS	PT1	PX1	PT0	PX0

Symbol	Position	Function
—	IP.7	reserved
—	IP.6	reserved
PT2	IP.5	defines the Timer 2 interrupt priority level. PT2 = 1 programs it to the higher priority level.
PS	IP.4	defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	defines the Timer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PX1	IP.2	defines the External Interrupt 1 priority level. PX1 = 1 programs it to the higher priority level.
PT0	IP.1	defines the Timer 0 interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	IP.0	defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

Figure 18. IP (Interrupt Priority) Register in the 8052AH

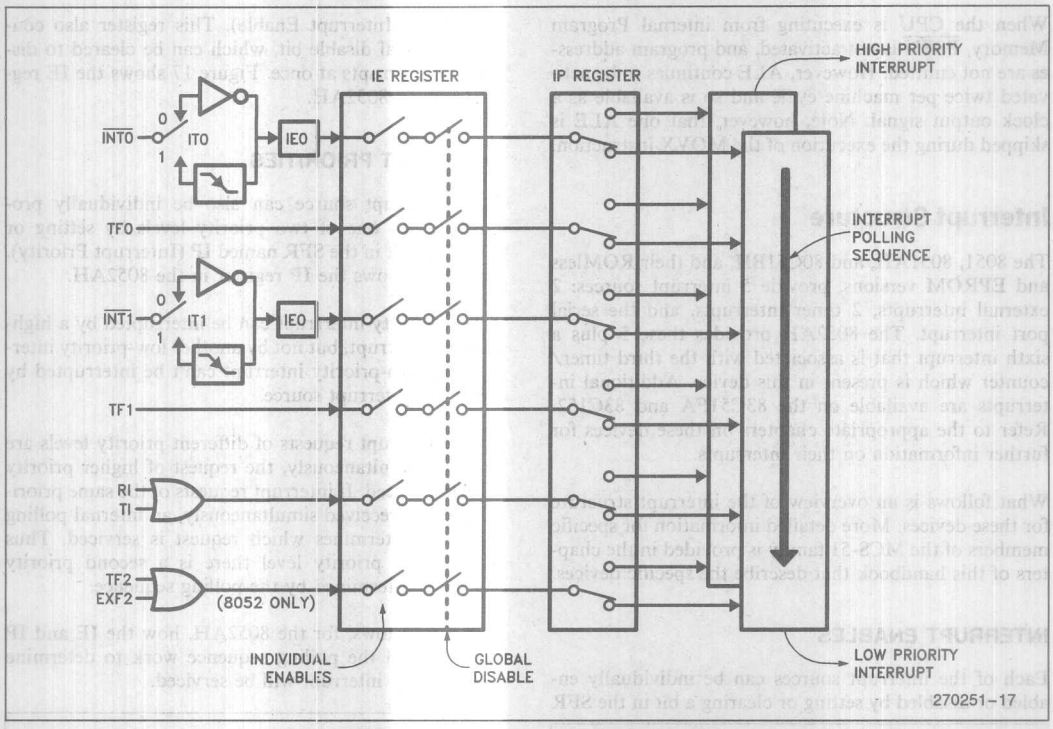


Figure 19. 8052 Interrupt Control System

In operation, all the interrupt flags are latched into the interrupt control system during State 5 of every machine cycle. The samples are polled during the following machine cycle. If the flag for an enabled interrupt is found to be set (1), the interrupt system generates an LCALL to the appropriate location in Program Memory, unless some other condition blocks the interrupt. Several conditions can block an interrupt, among them that an interrupt of equal or higher priority level is already in progress.

The hardware-generated LCALL causes the contents of the Program Counter to be pushed onto the stack, and reloads the PC with the beginning address of the service routine. As previously noted (Figure 3), the service routine for each interrupt begins at a fixed location.

Only the Program Counter is automatically pushed onto the stack, not the PSW or any other register. Having only the PC be automatically saved allows the programmer to decide how much time to spend saving which other registers. This enhances the interrupt response time, albeit at the expense of increasing the programmer's burden of responsibility. As a result, many interrupt functions that are typical in control applications—toggling a port pin, for example, or reloading a timer, or unloading a serial buffer—can often be completed in less time than it takes other architectures to commence them.

SIMULATING A THIRD PRIORITY LEVEL IN SOFTWARE

Some applications require more than the two priority levels that are provided by on-chip hardware in MCS-51 devices. In these cases, relatively simple software can be written to produce the same effect as a third priority level.

First, interrupts that are to have higher priority than 1 are assigned to priority 1 in the IP (Interrupt Priority) register. The service routines for priority 1 interrupts that are supposed to be interruptible by "priority 2" interrupts are written to include the following code:

```
        PUSH    IE
        MOV     IE,#MASK
        CALL    LABEL
          *******
(execute service routine)
          *******
        POP     IE
        RET
LABEL:  RETI
```

As soon as any priority 1 interrupt is acknowledged, the IE (Interrupt Enable) register is re-defined so as to disable all but "priority 2" interrupts. Then, a CALL to LABEL executes the RETI instruction, which clears the priority 1 interrupt-in-progress flip-flop. At this point any priority 1 interrupt that is enabled can be serviced, but only "priority 2" interrupts are enabled.

POPping IE restores the original enable byte. Then a normal RET (rather than another RETI) is used to terminate the service routine. The additional software adds 10 μs (at 12 MHz) to priority 1 interrupts.

Hardware Description of the 8051, 8052 and 80C51

6

6 Hardware Description of the 8051, 8052 and 80C

HARDWARE DESCRIPTION OF THE 8051, 8052 AND 80C51

INTRODUCTION

This chapter presents a comprehensive description of the on-chip hardware features of the MCS®-51 microcontrollers. Included in this description are

- The port drivers and how they function both as ports and, for Ports 0 and 2, in bus operations
- The Timer/Counters
- The Serial Interface
- The Interrupt System
- Reset
- The Reduced Power Modes in the CHMOS devices
- The EPROM versions of the 8051AH, 8052AH, and 80C51BH

The devices under consideration are listed in Table 1. As it becomes unwieldy to be constantly referring to each of these devices by their individual names, we will adopt a convention of referring to them generically as 8051s and 8052s, unless a specific member of the group is being referred to, in which case it will be specifically named. The "8051s" include the 8051, 8051AH, and 80C51BH, and their ROMless and EPROM versions. The "8052s" are the 8052AH, 8032AH, and 8752BH.

Figure 1 shows a functional block diagram of the 8051s and 8052s.

Table 1. The MCS-51 Family of Microcontrollers

Device Name	ROMless Version	EPROM Version	ROM Bytes	RAM Bytes	16-bit Timers	Ckt Type
8051	8031	(8751)	4K	128	2	HMOS
8051AH	8031AH	8751H	4K	128	2	HMOS
8052AH	8032AH	8752BH	8K	256	3	HMOS
80C51BH	80C31BH	87C51	4K	128	2	CHMOS

Special Function Registers

A map of the on-chip memory area called SFR (Special Function Register) space is shown in Figure 2. SFRs marked by parentheses are resident in the 8052s but not in the 8051s.

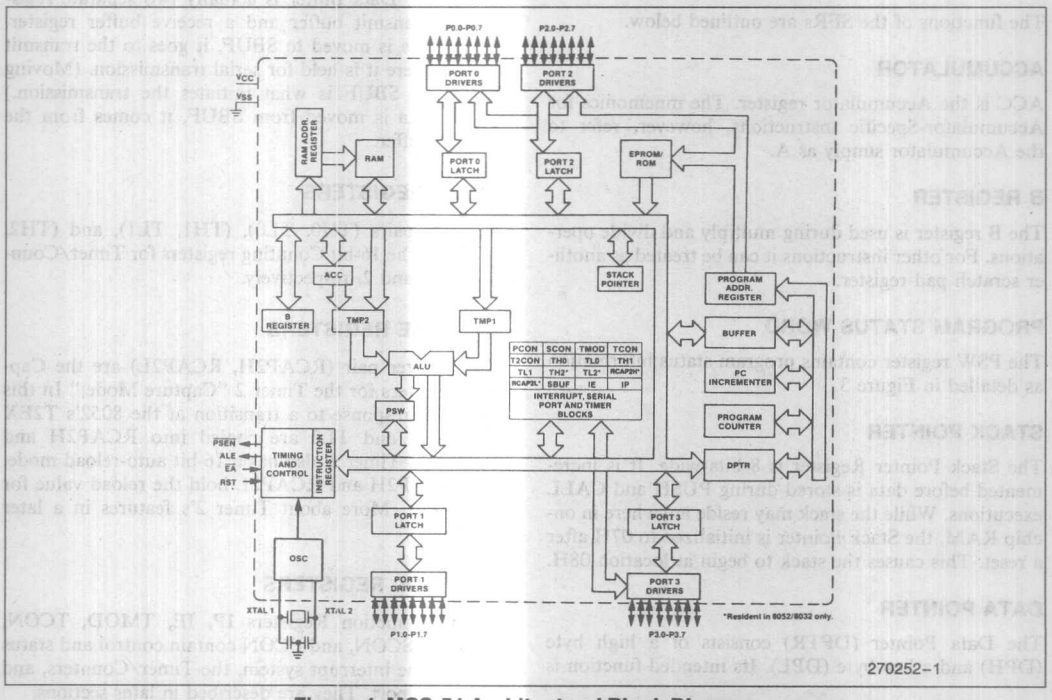


Figure 1. MCS-51 Architectural Block Diagram

	8 Bytes								
F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	(T2CON)		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

Figure 2. SFR Map. (. . .) Indicates Resident in 8052s, not in 8051s

Note that not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

User software should not write 1s to these unimplemented locations, since they may be used in future MCS-51 products to invoke new features. In that case the reset or inactive values of the new bits will always be 0, and their active values will be 1.

The functions of the SFRs are outlined below.

ACCUMULATOR

ACC is the Accumulator register. The mnemonics for Accumulator-Specific instructions, however, refer to the Accumulator simply as A.

B REGISTER

The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

PROGRAM STATUS WORD

The PSW register contains program status information as detailed in Figure 3.

STACK POINTER

The Stack Pointer Register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

DATA POINTER

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

PORTS 0 TO 3

P0, P1, P2 and P3 are the SFR latches of Ports 0, 1, 2 and 3, respectively.

SERIAL DATA BUFFER

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

TIMER REGISTERS

Register pairs (TH0, TL0), (TH1, TL1), and (TH2, TL2) are the 16-bit Counting registers for Timer/Counters 0, 1, and 2, respectively.

CAPTURE REGISTERS

The register pair (RCAP2H, RCAP2L) are the Capture registers for the Timer 2 "Capture Mode." In this mode, in response to a transition at the 8052's T2EX pin, TH2 and TL2 are copied into RCAP2H and RCAP2L. Timer 2 also has a 16-bit auto-reload mode, and RCAP2H and RCAP2L hold the reload value for this mode. More about Timer 2's features in a later section.

CONTROL REGISTERS

Special Function Registers IP, IE, TMOD, TCON, T2CON, SCON, and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port. They are described in later sections.

(MSB)							(LSB)
CY	AC	F0	RS1	RS0	OV	—	P

Symbol	Position	Name and Significance
CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary Carry flag. (For BCD operations.)
F0	PSW.5	Flag 0 (Available to the user for general purposes.)
RS1	PSW.4	Register bank select control bits 1 & 0. Set/cleared by software to determine working register bank (see Note).
RS0	PSW.3	
OV	PSW.2	Overflow flag.
—	PSW.1	User definable flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the Accumulator, i.e., even parity.

NOTE:
The contents of (RS1, RS0) enable the working register banks as follows:

(0.0)—Bank 0 (00H–07H)
(0.1)—Bank 1 (08H–0FH)
(1.0)—Bank 2 (10H–17H)
(1.1)—Bank 3 (18H–1FH)

Figure 3. PSW: Program Status Word Register

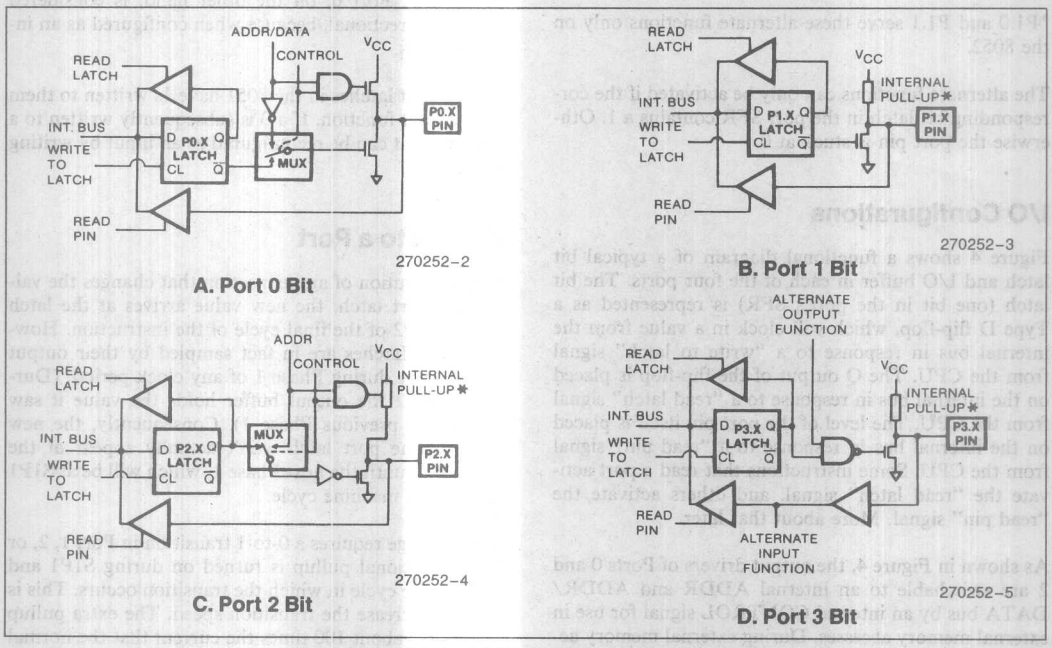


Figure 4. 8051 Port Bit Latches and I/O Buffers
*See Figure 5 for details of the internal pullup.

PORT STRUCTURES AND OPERATION

All four ports in the 8051 are bidirectional. Each consists of a latch (Special Function Registers P0 through P3), an output driver, and an input buffer.

The output drivers of Ports 0 and 2, and the input buffers of Port 0, are used in accesses to external memory. In this application, Port 0 outputs the low byte of the external memory address, time-multiplexed with the byte being written or read. Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise the Port 2 pins continue to emit the P2 SFR content.

All the Port 3 pins, and (in the 8052) two Port 1 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed on the following page.

Port Pin	Alternate Function
*P1.0	T2 (Timer/Counter 2 external input)
*P1.1	T2EX (Timer/Counter 2 Capture/Reload trigger)
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt)
P3.3	$\overline{INT1}$ (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	$\overline{WR}$ (external Data Memory write strobe)
P3.7	$\overline{RD}$ (external Data Memory read strobe)

*P1.0 and P1.1 serve these alternate functions only on the 8052.

The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise the port pin is stuck at 0.

I/O Configurations

Figure 4 shows a functional diagram of a typical bit latch and I/O buffer in each of the four ports. The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which will clock in a value from the internal bus in response to a "write to latch" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in response to a "read latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal. More about that later.

As shown in Figure 4, the output drivers of Ports 0 and 2 are switchable to an internal ADDR and ADDR/DATA bus by an internal CONTROL signal for use in external memory accesses. During external memory accesses, the P2 SFR remains unchanged, but the P0 SFR gets 1s written to it.

Also shown in Figure 4, is that if a P3 bit latch contains a 1, then the output level is controlled by the signal labeled "alternate output function." The actual P3.X pin level is always available to the pin's alternate input function, if any.

Ports 1, 2, and 3 have internal pullups. Port 0 has open drain outputs. Each I/O line can be independently used as an input or an output. (Ports 0 and 2 may not be used as general purpose I/O when being used as the ADDR/DATA BUS). To be used as an input, the port bit latch must contain a 1, which turns off the output driver FET. Then, for Ports 1, 2, and 3, the pin is pulled high by the internal pullup, but can be pulled low by an external source.

Port 0 differs in not having internal pullups. The pullup FET in the P0 output driver (see Figure 4) is used only when the Port is emitting 1s during external memory accesses. Otherwise the pullup FET is off. Consequently P0 lines that are being used as output port lines are open drain. Writing a 1 to the bit latch leaves both output FETs off, so the pin floats. In that condition it can be used a high-impedance input.

Because Ports 1, 2, and 3 have fixed internal pullups they are sometimes called "quasi-bidirectional" ports. When configured as inputs they pull high and will source current (IIL, in the data sheets) when externally pulled low. Port 0, on the other hand, is considered "true" bidirectional, because when configured as an input it floats.

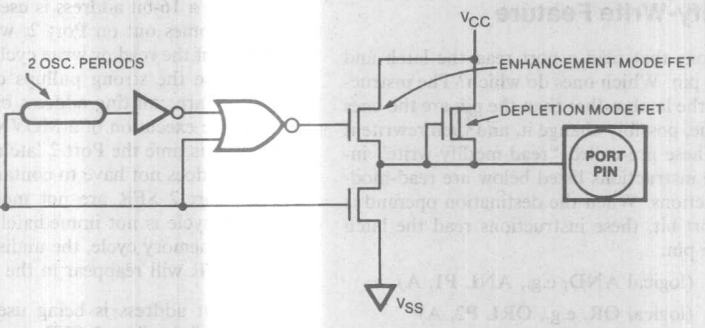
All the port latches in the 8051 have 1s written to them by the reset function. If a 0 is subsequently written to a port latch, it can be reconfigured as an input by writing a 1 to it.

Writing to a Port

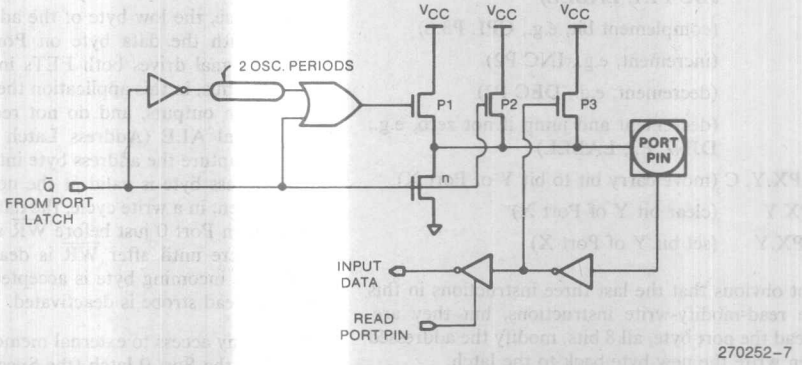
In the execution of an instruction that changes the value in a port latch, the new value arrives at the latch during S6P2 of the final cycle of the instruction. However, port latches are in fact sampled by their output buffers only during Phase 1 of any clock period. (During Phase 2 the output buffer holds the value it saw during the previous Phase 1). Consequently, the new value in the port latch won't actually appear at the output pin until the next Phase 1, which will be at S1P1 of the next machine cycle.

If the change requires a 0-to-1 transition in Port 1, 2, or 3, an additional pullup is turned on during S1P1 and S1P2 of the cycle in which the transition occurs. This is done to increase the transition speed. The extra pullup can source about 100 times the current that the normal pullup can. It should be noted that the internal pullups are field-effect transistors, not linear resistors. The pullup arrangements are shown in Figure 5.

In HMOS versions of the 8051, the fixed part of the pullup is a depletion-mode transistor with the gate wired to the source. This transistor will allow the pin to source about 0.25 mA when shorted to ground. In parallel with the fixed pullup is an enhancement-mode transistor, which is activated during S1 whenever the port bit does a 0-to-1 transition. During this interval, if the port pin is shorted to ground, this extra transistor will allow the pin to source an additional 30 mA.



A. HMOS Configuration. The enhancement mode transistor is turned on for 2 osc. periods after Q makes a 1-to-0 transition.



B. CHMOS Configuration. pFET 1 is turned on for 2 osc. periods after Q makes a 1-to-0 transition. During this time, pFET 1 also turns on pFET 3 through the inverter to form a latch which holds the 1. pFET 2 is also on.

Figure 5. Ports 1 And 3 HMOS And CHMOS Internal Pullup Configurations. Port 2 is Similar Except That It Holds The Strong Pullup On While Emitting 1s That Are Address Bits. (See Text, "Accessing External Memory".)

In the CHMOS versions, the pullup consists of three pFETs. It should be noted that an n-channel FET (nFET) is turned on when a logical 1 is applied to its gate, and is turned off when a logical 0 is applied to its gate. A p-channel FET (pFET) is the opposite: it is on when its gate sees a 0, and off when its gate sees a 1.

pFET1 in Figure 5 is the transistor that is turned on for 2 oscillator periods after a 0-to-1 transition in the port latch. While it's on, it turns on pFET3 (a weak pull-up), through the inverter. This inverter and pFET form a latch which hold the 1.

Note that if the pin is emitting a 1, a negative glitch on the pin from some external source can turn off pFET3, causing the pin to go into a float state. pFET2 is a very weak pullup which is on whenever the nFET is off, in traditional CMOS style. It's only about $1/10$ the strength of pFET3. Its function is to restore a 1 to the pin in the event the pin *had* a 1 and lost it to a glitch.

Port Loading and Interfacing

The output buffers of Ports 1, 2, and 3 can each drive 4 LS TTL inputs. These ports on HMOS versions can be driven in a normal manner by any TTL or NMOS circuit. Both HMOS and CHMOS pins can be driven by open-collector and open-drain outputs, but note that 0-to-1 transitions will not be fast. In the HMOS device, if the pin is driven by an open-collector output, a 0-to-1 transition will have to be driven by the relatively weak depletion mode FET in Figure 5(A). In the CHMOS device, an input 0 turns off pullup pFET3, leaving only the very weak pullup pFET2 to drive the transition.

Port 0 output buffers can each drive 8 LS TTL inputs. They do, however, require external pullups to drive NMOS inputs, except when being used as the ADDRESS/DATA bus.

Read-Modify-Write Feature

Some instructions that read a port read the latch and others read the pin. Which ones do which? The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write" instructions. The instructions listed below are read-modify-write instructions. When the destination operand is a port, or a port bit, these instructions read the latch rather than the pin:

ANL	(logical AND, e.g., ANL P1, A)
ORL	(logical OR, e.g., ORL P2, A)
XRL	(logical EX-OR, e.g., XRL P3, A)
JBC	(jump if bit = 1 and clear bit, e.g., JBC P1.1, LABEL)
CPL	(complement bit, e.g., CPL P3.0)
INC	(increment, e.g., INC P2)
DEC	(decrement, e.g., DEC P2)
DJNZ	(decrement and jump if not zero, e.g., DJNZ P3, LABEL)
MOV, PX.Y, C	(move carry bit to bit Y of Port X)
CLR PX.Y	(clear bit Y of Port X)
SETB PX.Y	(set bit Y of Port X)

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of a transistor. When a 1 is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as a 0. Reading the latch rather than the pin will return the correct value of 1.

ACCESSING EXTERNAL MEMORY

Accesses to external memory are of two types: accesses to external Program Memory and accesses to external Data Memory. Accesses to external Program Memory use signal $\overline{\text{PSEN}}$ (program store enable) as the read strobe. Accesses to external Data Memory use $\overline{\text{RD}}$ or $\overline{\text{WR}}$ (alternate functions of P3.7 and P3.6) to strobe the memory.

Fetches from external Program Memory always use a 16-bit address. Accesses to external Data Memory can use either a 16-bit address (MOVX @DPTR) or an 8-bit address (MOVX @Ri).

Whenever a 16-bit address is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle. Note that the Port 2 drivers use the strong pullups during the entire time that they are emitting address bits that are 1s. This is during the execution of a MOVX @DPTR instruction. During this time the Port 2 latch (the Special Function Register) does not have to contain 1s, and the contents of the Port 2 SFR are not modified. If the external memory cycle is not immediately followed by another external memory cycle, the undisturbed contents of the Port 2 SFR will reappear in the next cycle.

If an 8-bit address is being used (MOVX @Ri), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging.

In any case, the low byte of the address is time-multiplexed with the data byte on Port 0. The ADDR/DATA signal drives both FETs in the Port 0 output buffers. Thus, in this application the Port 0 pins are not open-drain outputs, and do not require external pullups. Signal ALE (Address Latch Enable) should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before $\overline{\text{WR}}$ is activated, and remains there until after $\overline{\text{WR}}$ is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe is deactivated.

During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating whatever information the Port 0 SFR may have been holding.

External Program Memory is accessed under two conditions:

1) Whenever signal $\overline{\text{EA}}$ is active; or
2) Whenever the program counter (PC) contains a number that is larger than 0FFFH (1FFFH for the 8052).

This requires that the ROMless versions have $\overline{\text{EA}}$ wired low to enable the lower 4K (8K for the 8032) program bytes to be fetched from external memory.

When the CPU is executing out of external Program Memory, all 8 bits of Port 2 are dedicated to an output function and may not be used for general purpose I/O. During external program fetches they output the high byte of the PC. During this time the Port 2 drivers use the strong pullups to emit PC bits that are 1s.

TIMER/COUNTERS

The 8051 has two 16-bit Timer/Counter registers: Timer 0 and Timer 1. The 8052 has these two plus one more: Timer 2. All three can be configured to operate either as timers or event counters.

In the "Timer" function, the register is incremented every machine cycle. Thus, one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is $1/12$ of the oscillator frequency.

In the "Counter" function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0, T1 or (in the 8052) T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is $1/24$ of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full machine cycle.

In addition to the "Timer" or "Counter" selection, Timer 0 and Timer 1 have four operating modes from which to select. Timer 2, in the 8052, has three modes of operation: "Capture," "Auto-Reload" and "baud rate generator."

Timer 0 and Timer 1

These Timer/Counters are present in both the 8051 and the 8052. The "Timer" or "Counter" function is selected by control bits C/$\overline{T}$ in the Special Function Register TMOD (Figure 6). These two Timer/Counters have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for both Timer/Counters. Mode 3 is different. The four operating modes are described in the following text.

MODE 0

Putting either Timer into Mode 0 makes it look like an 8048 Timer, which is an 8-bit Counter with a divide-by-32 prescaler. Figure 7 shows the Mode 0 operation as it applies to Timer 1.

In this mode, the Timer register is configured as a 13-Bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or $\overline{INT1}$ = 1. (Setting GATE = 1 allows the Timer to be controlled by external input $\overline{INT1}$, to facilitate pulse width measurements.) TR1 is a control bit in the Special Function Register TCON (Figure 8). GATE is in TMOD.

The 13-Bit register consists of all 8 bits of TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored. Setting the run flag (TR1) does not clear the registers.

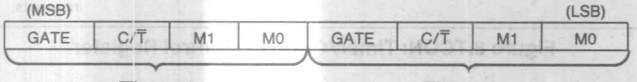
Mode 0 operation is the same for Timer 0 as for Timer 1. Substitute TR0, TF0 and $\overline{INT0}$ for the corresponding Timer 1 signals in Figure 7. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

MODE 1

Mode 1 is the same as Mode 0, except that the Timer register is being run with all 16 bits.

MODE 2

Mode 2 configures the Timer register as an 8-bit Counter (TL1) with automatic reload, as shown in Figure 9. Overflow from TL1 not only sets TF1, but also reloads



GATE	Gating control when set. Timer/Counter "x" is enabled only while "$\overline{INTx}$" pin is high and "TRx" control pin is set. When cleared Timer "x" is enabled whenever "TRx" control bit is set.
C/$\overline{T}$	Timer or Counter Selector cleared for Timer operation (input from internal system clock). Set for Counter operation (input from "Tx" input pin).

M1	M0	Operating Mode
0	0	MCS-48 Timer "TLx" serves as 5-bit prescaler.
0	1	16-bit Timer/Counter "THx" and "TLx" are cascaded; there is no prescaler.
1	0	8-bit auto-reload Timer/Counter "THx" holds a value which is to be reloaded into "TLx" each time it overflows.
1	1	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits.
1	1	(Timer 1) Timer/Counter 1 stopped.

Figure 6. TMOD: Timer/Counter Mode Control Register

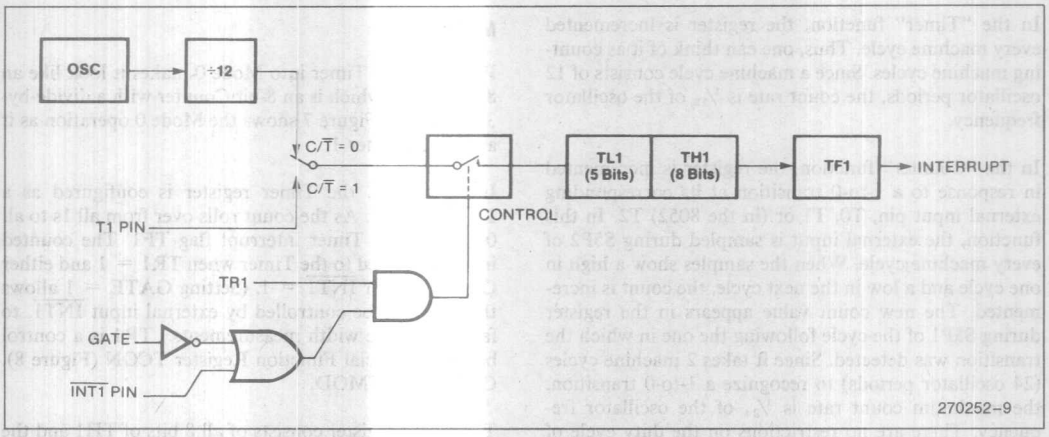


Figure 7. Timer/Counter 1 Mode 0: 13-Bit Counter

(MSB)							(LSB)
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors tc interrupt routine.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TF0	TCON.5	Timer 0 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors tc interrupt routine.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

Figure 8.TCON: Timer/Counter Control Register

TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.

Mode 2 operation is the same for Timer/Counter 0.

MODE 3

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0.

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 10. TL0 uses the Timer 0 control bits: C/$\overline{T}$, GATE, TR0, $\overline{INT0}$, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the "Timer 1" interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer or counter. With Timer 0 in Mode 3, an 8051 can look like it has three Timer/Counters, and an 8052, like it has four. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3, or can still be used by the serial port as a baud rate generator, or in fact, in any application not requiring an interrupt.

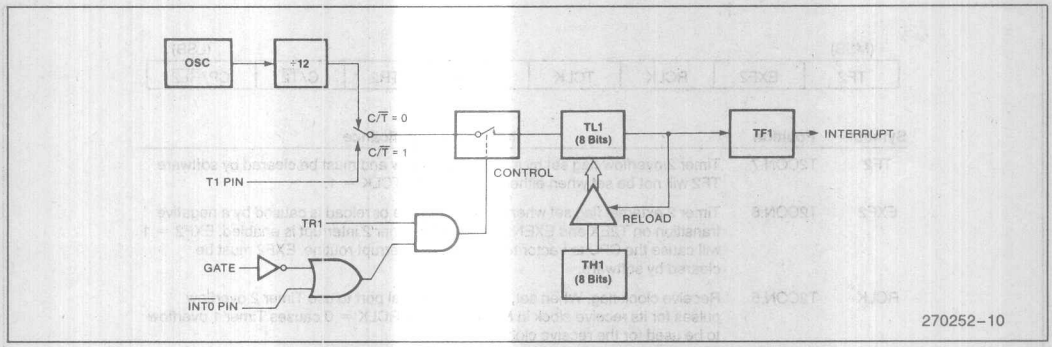


Figure 9. Timer/Counter 1 Mode 2: 8-Bit Auto-Reload

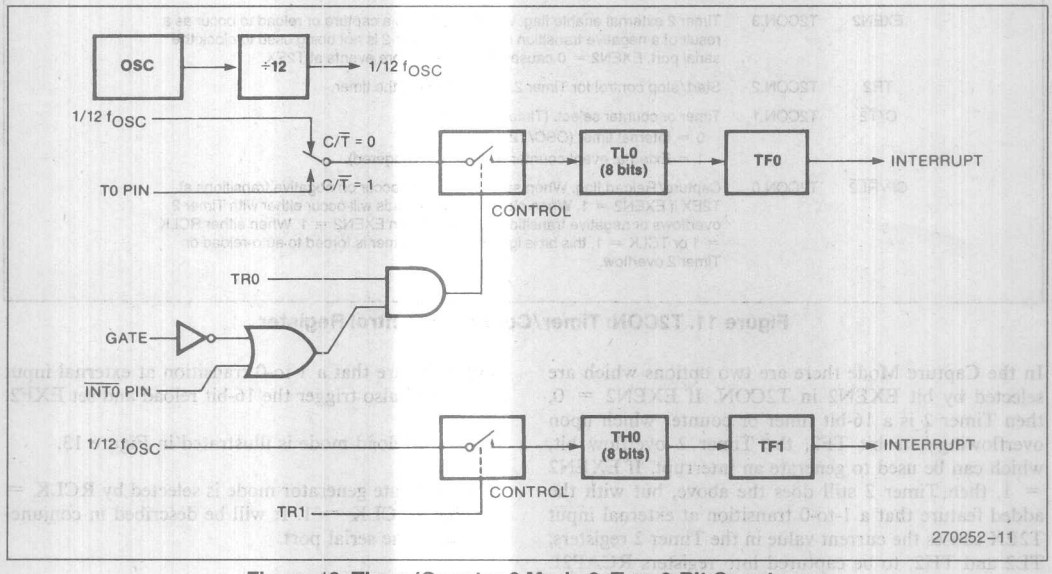


Figure 10. Timer/Counter 0 Mode 3: Two 8-Bit Counters

Timer 2

Timer 2 is a 16-bit Timer/Counter which is present only in the 8052. Like Timers 0 and 1, it can operate either as a timer or as an event counter. This is selected by bit C/$\overline{T2}$ in the Special Function Register T2CON (Figure 11). It has three operating modes: "capture," "auto-load" and "baud rate generator," which are selected by bits in T2CON as shown in Table 2.

Table 2. Timer 2 Operating Modes

RCLK + TCLK	CP/$\overline{RL2}$	TR2	Mode
0	0	1	16-bit Auto-Reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(off)

(MSB)							(LSB)
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

Symbol	Position	Name and Significance
TF2	T2CON.7	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	T2CON.6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.
RCLK	T2CON.5	Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	T2CON.4	Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	T2CON.3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	T2CON.2	Start/stop control for Timer 2. A logic 1 starts the timer.
C/$\overline{T2}$	T2CON.1	Timer or counter select. (Timer 2) 0 = Internal timer (OSC/12) 1 = External event counter (falling edge triggered).
CP/$\overline{RL2}$	T2CON.0	Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Figure 11. T2CON: Timer/Counter 2 Control Register

In the Capture Mode there are two options which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then Timer 2 is a 16-bit timer or counter which upon overflowing sets bit TF2, the Timer 2 overflow bit, which can be used to generate an interrupt. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in the Timer 2 registers, TL2 and TH2, to be captured into registers RCAP2L and RCAP2H, respectively. (RCAP2L and RCAP2H are new Special Function Registers in the 8052.) In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and EXF2, like TF2, can generate an interrupt.

The Capture Mode is illustrated in Figure 12.

In the auto-reload mode there are again two options, which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then when Timer 2 rolls over it not only sets TF2 but also causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2L and RCAP2H, which are preset by software. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX will also trigger the 16-bit reload and set EXF2.

The auto-reload mode is illustrated in Figure 13.

The baud rate generator mode is selected by RCLK = 1 and/or TCLK = 1. It will be described in conjunction with the serial port.

SERIAL INTERFACE

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register. (However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost). The serial port receive and transmit registers are both accessed at Special Function Register SBUF. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

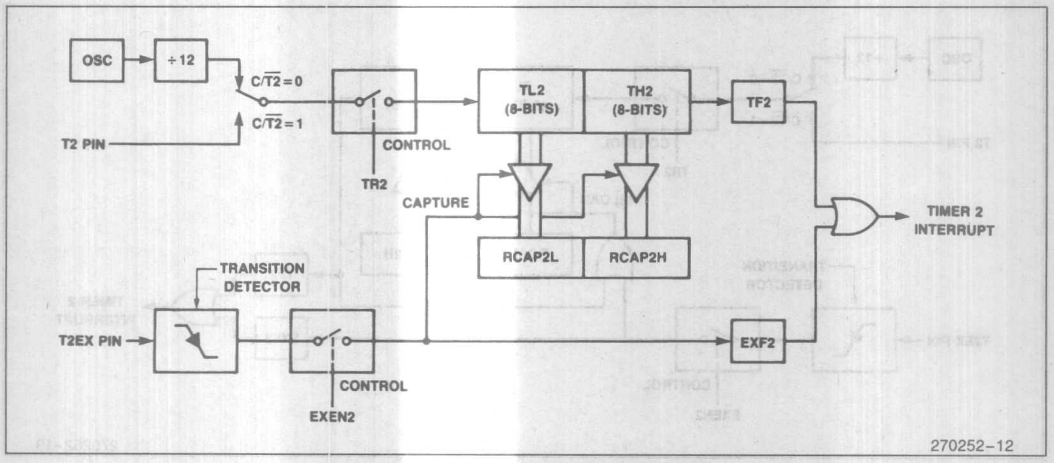


Figure 12. Timer 2 in Capture Mode

The serial port can operate in 4 modes:

Mode 0: Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at 1/12 the oscillator frequency.

Mode 1: 10 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in Special Function Register SCON. The baud rate is variable.

Mode 2: 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On Transmit, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On receive, the 9th data bit goes into RB8 in Special Functon Register SCON, while the stop bit is ignored. The baud rate is programmable to either $1/32$ or $1/64$ the oscillator frequency.

Mode 3: 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except the baud rate. The baud rate in Mode 3 is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1.

Multiprocessor Communications

Modes 2 and 3 have a special provision for multiprocessor communications. In these modes, 9 data bits are received. The 9th one goes into RB8. Then comes a stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8 = 1. This feature is enabled by setting bit SM2 in SCON. A way to use this feature in multiprocessor systems is as follows.

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM2s set and go on about their business, ignoring the coming data bytes.

SM2 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. In a Mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

Serial Port Control Register

The serial port control and status register is the Special Function Register SCON, shown in Figure 14. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8), and the serial port interrupt bits (TI and RI).

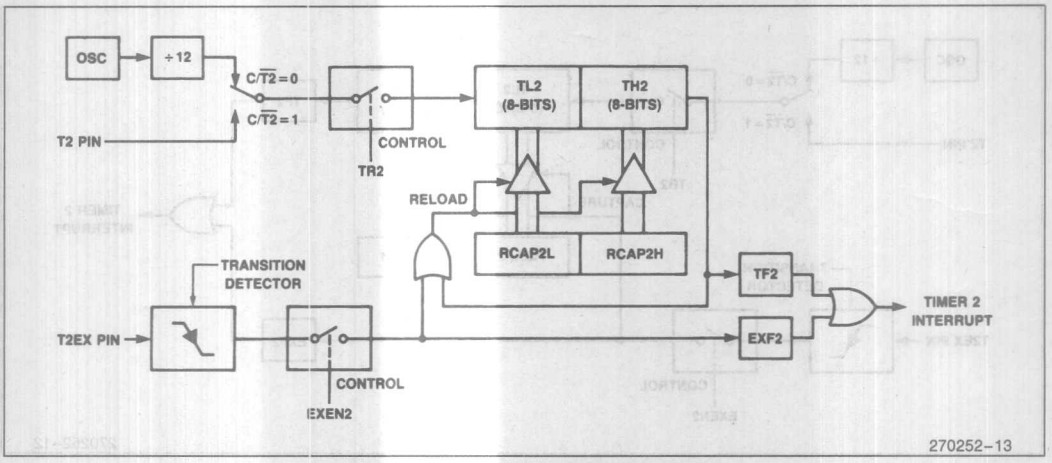


Figure 13. Timer 2 in Auto-Reload Mode

(MSB)							(LSB)
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Where SM0, SM1 specify the serial port mode, as follows:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	shift register	$f_{osc.}/12$
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	$f_{osc.}/64$ or $f_{osc.}/32$
1	1	3	9-bit UART	variable

- SM2 enables the multiprocessor communication feature in Modes 2 and 3. In Mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In Mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In Mode 0, SM2 should be 0.
- REN enables serial reception. Set by software to enable reception. Clear by software to disable reception.
- TB8 is the 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.
- RB8 in Modes 2 and 3, is the 9th data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.
- TI is transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.
- RI is receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

Figure 14. SCON: Serial Port Control Register

Baud Rates

The baud rate in Mode 0 is fixed:

$$\text{Mode 0 Baud Rate} = \frac{\text{Oscillator Frequency}}{12}$$

The baud rate in Mode 2 depends on the value of bit SMOD in Special Function Register PCON. If SMOD = 0 (which is the value on reset), the baud rate $1/64$ the oscillator frequency. If SMOD = 1, the baud rate is $1/32$ the oscillator frequency.

$$\text{Mode 2 Baud Rate} = \frac{2^{\text{SMOD}}}{64} \times (\text{Oscillator Frequency})$$

In the 8051, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate. In the 8052, these baud rates can be determined by Timer 1, or by Timer 2, or by both (one for transmit and the other for receive).

Using Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

$$\text{Modes 1, 3 Baud Rate} = \frac{2^{SMOD}}{32} \times \text{(Timer 1 Overflow Rate)}$$

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either "timer" or "counter" operation, and in any of its 3 running modes. In the most typical applications, it is configured for "timer" operation, in the auto-reload mode (high nibble of TMOD = 0010B). In that case, the baud rate is given by the formula

$$\text{Modes 1, 3 Baud Rate} = \frac{2^{SMOD}}{32} \times \frac{\text{Oscillator Frequency}}{12x\,[256 - (TH1)]}$$

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the Timer 1 interrupt to do a 16-bit software reload.

Figure 15 lists various commonly used baud rates and how they can be obtained from Timer 1.

Baud Rate	f_{OSC}	SMOD	Timer 1		
			C/$\overline{T}$	Mode	Reload Value
Mode 0 Max: 1 MHZ	12 MHZ	X	X	X	X
Mode 2 Max: 375K	12 MHZ	1	X	X	X
Modes 1, 3: 62.5K	12 MHZ	1	0	2	FFH
19.2K	11.059 MHZ	1	0	2	FDH
9.6K	11.059 MHZ	0	0	2	FDH
4.8K	11.059 MHZ	0	0	2	FAH
2.4K	11.059 MHZ	0	0	2	F4H
1.2K	11.059 MHZ	0	0	2	E8H
137.5K	11.986 MHZ	0	0	2	1DH
110K	6 MHZ	0	0	2	72H
110K	12 MHZ	0	0	1	FEEBH

Figure 15. Timer 1 Generated Commonly Used Baud Rates

Using Timer 2 to Generate Baud Rates

In the 8052, Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Figure 11). Note then the baud rates for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 16.

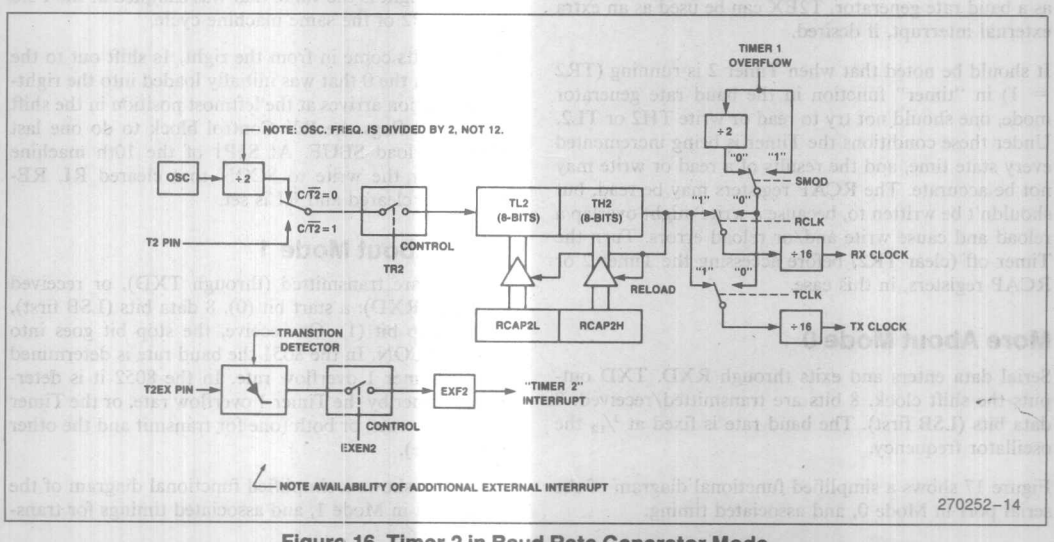


Figure 16. Timer 2 in Baud Rate Generator Mode

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

Now, the baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate as follows:

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either "timer" or "counter" operation. In the most typical applications, it is configured for "timer" operation (C/T2 = 0). "Timer" operation is a little different for Timer 2 when it's being used as a baud rate generator. Normally, as a timer it would increment every machine cycle (thus at $1/12$ the oscillator frequency). As a baud rate generator, however, it increments every state time (thus at $1/2$ the oscillator frequency). In that case the baud rate is given by the formula

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Oscillator Frequency}}{32\text{x}\,[65536 - (\text{RCAP2H, RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 16. This Figure is valid only if RCLK + TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Therefore, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

It should be noted that when Timer 2 is running (TR2 = 1) in "timer" function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the Timer is being incremented every state time, and the results of a read or write may not be accurate. The RCAP registers may be read, but shouldn't be written to, because a write might overlap a reload and cause write and/or reload errors. Turn the Timer off (clear TR2) before accessing the Timer 2 or RCAP registers, in this case.

More About Mode 0

Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at $1/12$ the oscillator frequency.

Figure 17 shows a simplified functional diagram of the serial port in Mode 0, and associated timing.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal at S6P2 also loads a 1 into the 9th position of the transmit shift register and tells the TX Control block to commence a transmission. The internal timing is such that one full machine cycle will elapse between "write to SBUF," and activation of SEND.

SEND enables the output of the shift register to the alternate output function line of P3.0, and also enables SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK is low during S3, S4, and S5 of every machine cycle, and high during S6, S1 and S2. At S6P2 of every machine cycle in which SEND is active, the contents of the transmit shift register are shifted to the right one position.

As data bits shift out to the right, zeroes come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position, is just to the left of the MSB, and all positions to the left of that contain zeroes. This condition flags the TX Control block to do one last shift and then deactivate SEND and set TI. Both of these actions occur at S1P1 of the 10th machine cycle after "write to SBUF."

Reception is initiated by the condition REN = 1 and RI = 0. At S6P2 of the next machine cycle, the RX Control unit writes the bits 11111110 to the receive shift register, and in the next clock phase activates RECEIVE.

RECEIVE enables SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK makes transitions at S3P1 and S6P1 of every machine cycle. At S6P2 of every machine cycle in which RECEIVE is active, the contents of the receive shift register are shifted to the left one position. The value that comes in from the right is the value that was sampled at the P3.0 pin at S5P2 of the same machine cycle.

As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX Control block to do one last shift and load SBUF. At S1P1 of the 10th machine cycle after the write to SCON that cleared RI, RECEIVE is cleared and RI is set.

More About Mode 1

Ten bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in SCON. In the 8051 the baud rate is determined by the Timer 1 overflow rate. In the 8052 it is determined either by the Timer 1 overflow rate, or the Timer 2 overflow rate, or both (one for transmit and the other for receive).

Figure 18 shows a simplified functional diagram of the serial port in Mode 1, and associated timings for transmit receive.

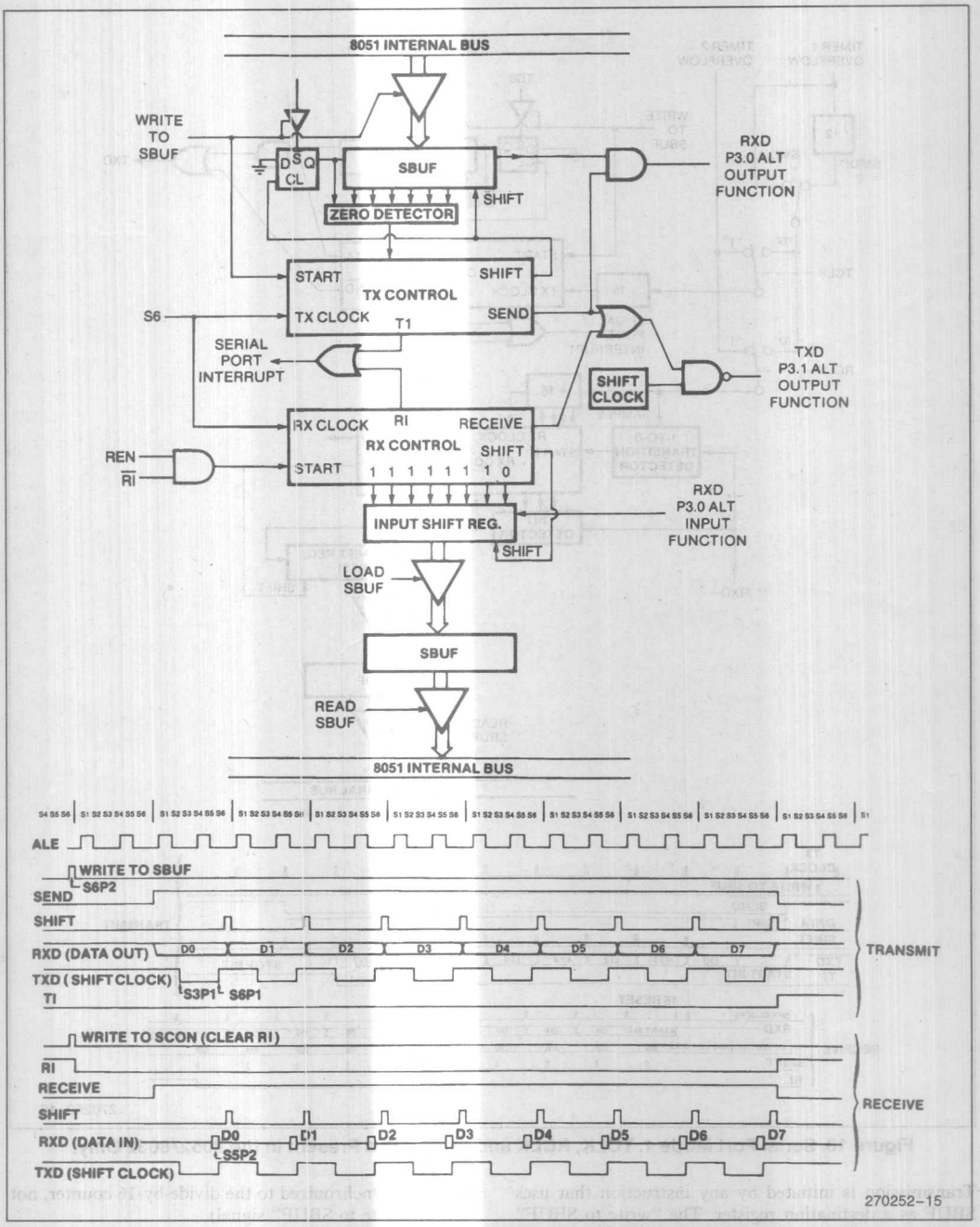


Figure 17. Serial Port Mode 0

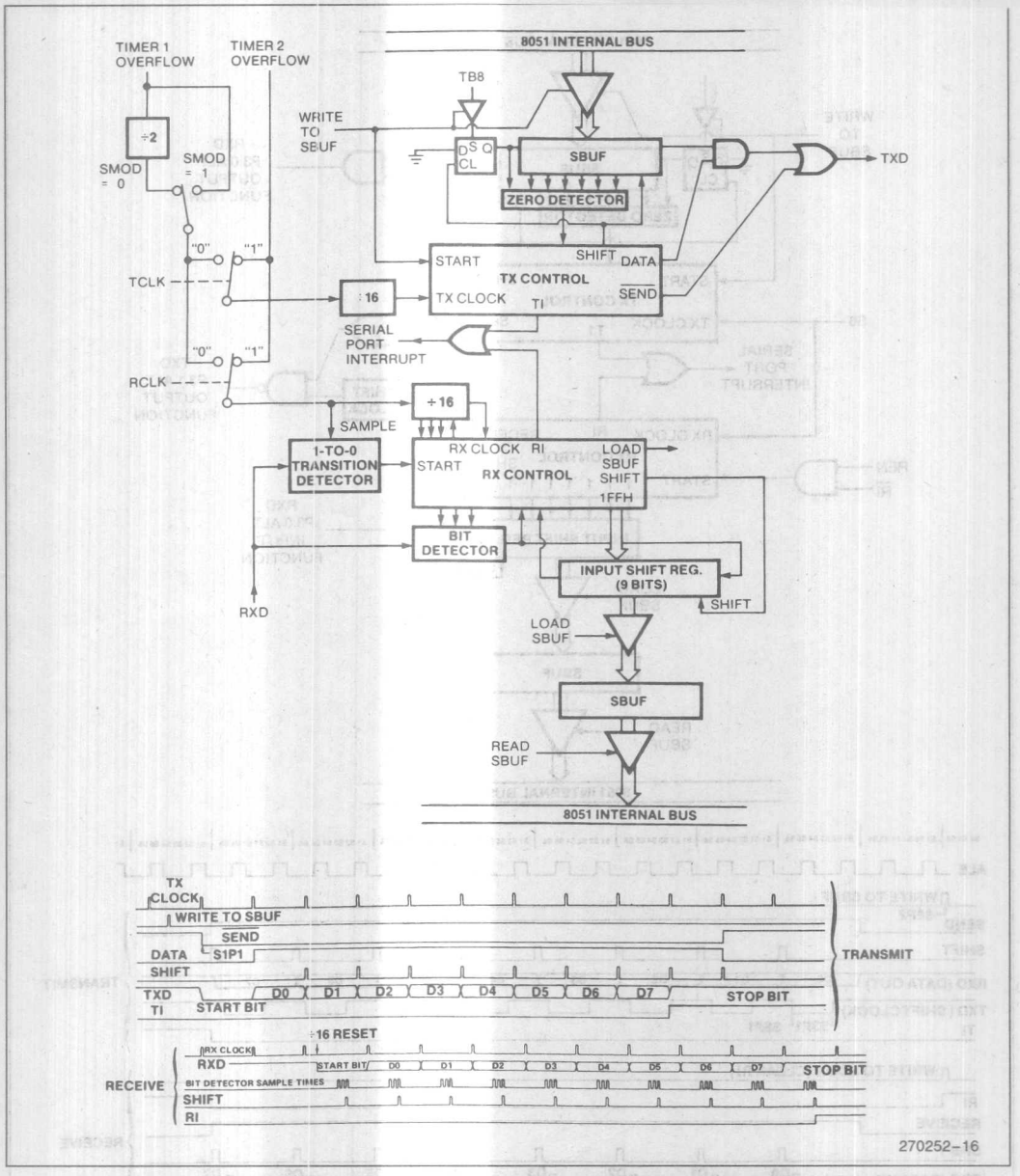


Figure 18. Serial Port Mode 1. TCLK, RCLK and Timer 2 are Present in the 8052/8032 Only.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads a 1 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission actually commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal).

The transmission begins with activation of $\overline{\text{SEND}}$, which puts the start bit at TXD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that.

As data bits shift out to the right, zeroes are clocked in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position is just to the left of the MSB, and all positions to the left of that contain zeroes. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI. This occurs at the 10th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written into the input shift register. Resetting the divide-by-16 counter aligns its rollovers with the boundaries of the incoming bit times.

The 16 states of the counter divide each bit time into 16ths. At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register, (which in mode 1 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated.

1) RI = 0, and
2) Either SM2 = 0, or the received stop bit = 1

If either of these two conditions is not met, the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB8, the 8 data bits go into SBUF, and RI is activated. At this time, whether the above conditions are met or not, the unit goes back to looking for a 1-to-0 transition in RXD.

More About Modes 2 and 3

Eleven bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmit, the 9th data bit (TB8) can be assigned the value of 0 or 1. On receive, the 9th data bit goes into RB8 in SCON. The baud rate is programmable to either $1/32$ or $1/64$ the oscillator frequency in Mode 2. Mode 3 may have a variable baud rate generated from either Timer 1 or 2 depending on the state of TCLK and RCLK.

Figures 19 and 20 show a functional diagram of the serial port in Modes 2 and 3. The receive portion is exactly the same as in Mode 1. The transmit portion differs from Mode 1 only in the 9th bit of the transmit shift register.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads TB8 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal.)

The transmission begins with activation of SEND, which puts the start bit at TXD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeroes are clocked in. Thus, as data bits shift out to the right, zeroes are clocked in from the left. When TB8 is at the output position of the shift register, then the stop bit is just to the left of TB8, and all positions to the left of that contain zeroes. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI. This occurs at the 11th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written to the input shift register.

At the 7th, 8th and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

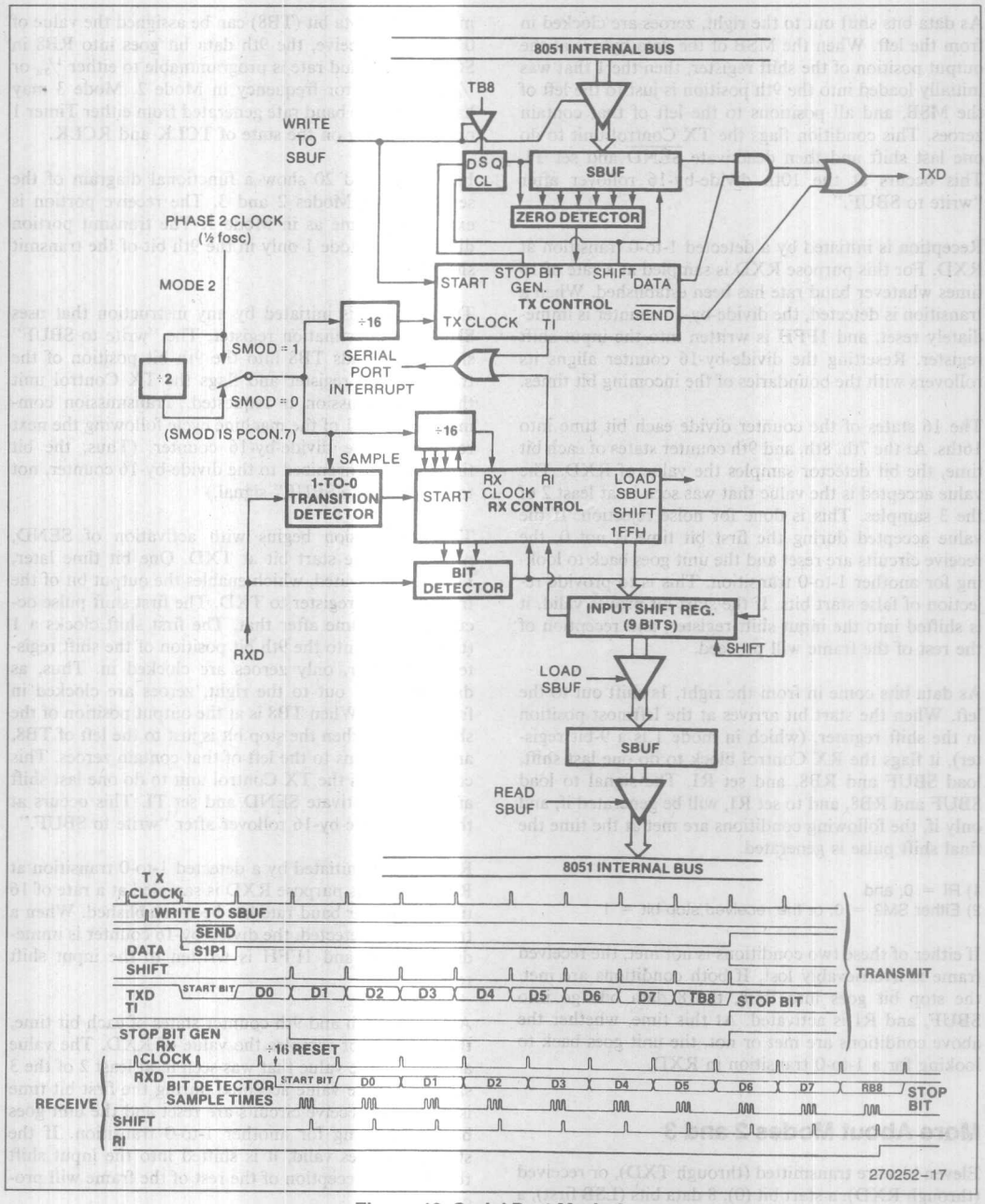


Figure 19. Serial Port Mode 2

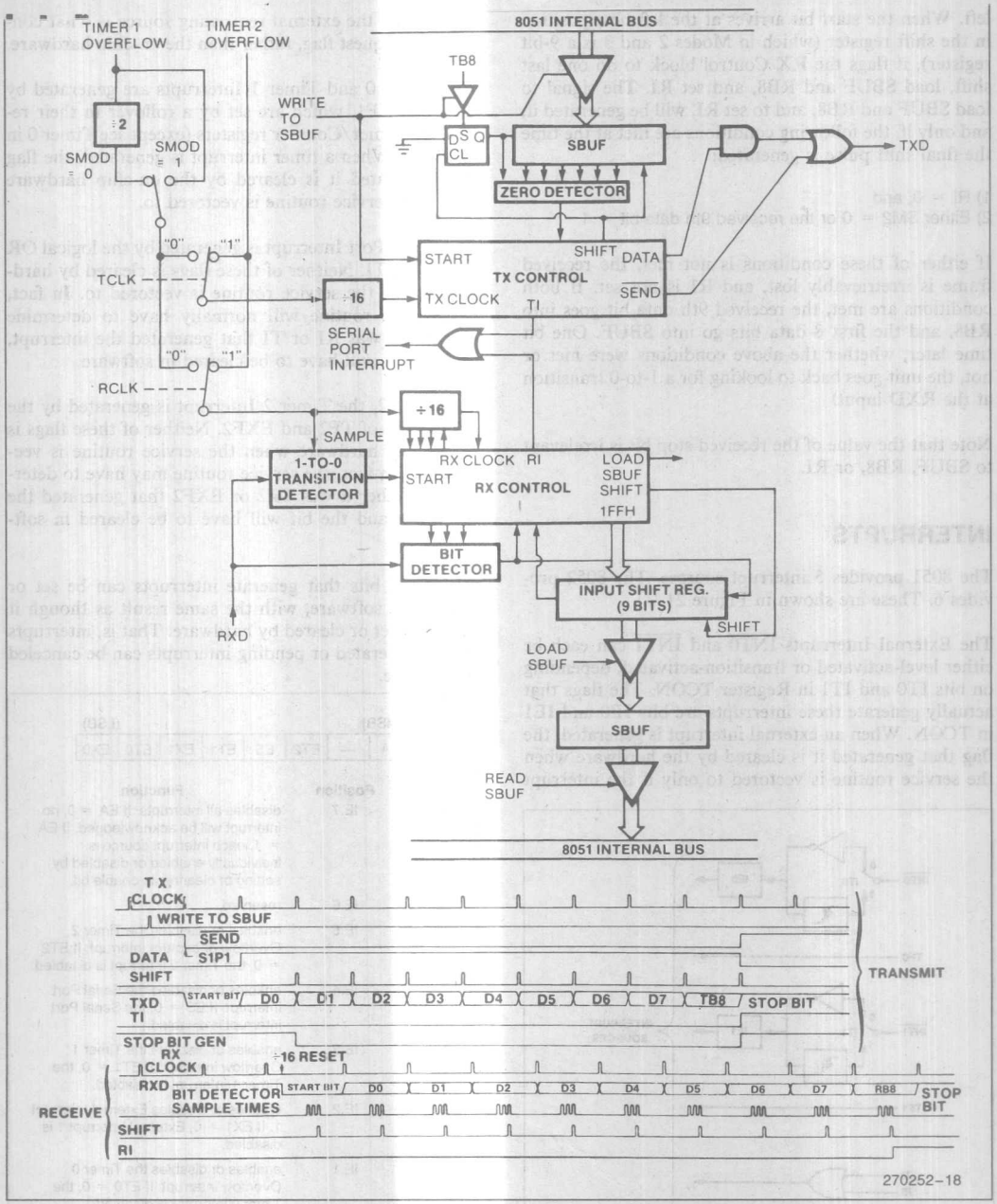


Figure 20. Serial Port Mode 3. TCLK, RCLK, and Timer 2 are Present in the 8052/8032 Only.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in Modes 2 and 3 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

1) RI = 0, and
2) Either SM2 = 0 or the received 9th data bit = 1

If either of these conditions is not met, the received frame is irretrievably lost, and RI is not set. If both conditions are met, the received 9th data bit goes into RB8, and the first 8 data bits go into SBUF. One bit time later, whether the above conditions were met or not, the unit goes back to looking for a 1-to-0 transition at the RXD input.

Note that the value of the received stop bit is irrelevant to SBUF, RB8, or RI.

INTERRUPTS

The 8051 provides 5 interrupt sources. The 8052 provides 6. These are shown in Figure 21.

The External Interrupts $\overline{INT0}$ and $\overline{INT1}$ can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in Register TCON. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt was transition-activated. If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

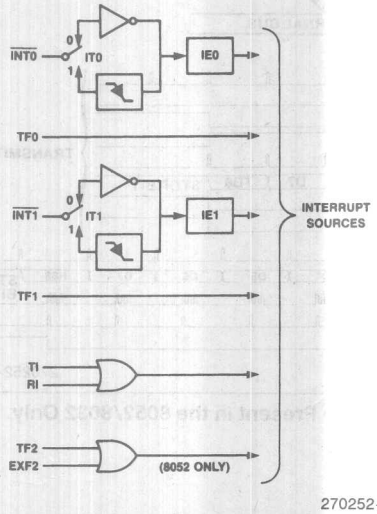


Figure 21. MCS®-51 Interrupt Sources

The Timer 0 and Timer 1 Interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter registers (except see Timer 0 in Mode 3). When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The Serial Port Interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine will normally have to determine whether it was RI or TI that generated the interrupt, and the bit will have to be cleared in software.

In the 8052, the Timer 2 Interrupt is generated by the logical OR of TF2 and EXF2. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared in software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

(MSB)							(LSB)
EA	—	ET2	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	reserved.
ET2	IE.5	enables or disables the Timer 2 Overflow or capture interrupt. If ET2 = 0, the Timer 2 interrupt is disabled.
ES	IE.4	enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	enables or disables External Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

Figure 22. IE: Interrupt Enable Register

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE (Figure 22). IE contains also a global disable bit, EA, which disables all interrupts at once.

Note in Figure 23 that bit position IE.6 is unimplemented. In the 8051s, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future MCS-51 products.

Priority Level Structure

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in Special Function Register IP (Figure 23). A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

(MSB) (LSB)

—	—	PT2	PS	PT1	PX1	PT0	PX0

Symbol	Position	Function
—	IP.7	reserved
—	IP.6	reserved
PT2	IP.5	defines the Timer 2 interrupt priority level. PT2 = 1 programs it to the higher priority level.
PS	IP.4	defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	defines the Timer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PT0	IP.1	defines the Timer 0 interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	IP.0	defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

Figure 23. IP: Interrupt Priority Register

If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the **same** priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence, as follows:

	Source	Priority Within Level
1.	IE0	(highest)
2.	TF0	
3.	IE1	
4.	TF1	
5.	RI + TI	
6.	TF2 + EXF2	(lowest)

Note that the "priority within level" structure is only used to resolve *simultaneous requests of the same priority level.*

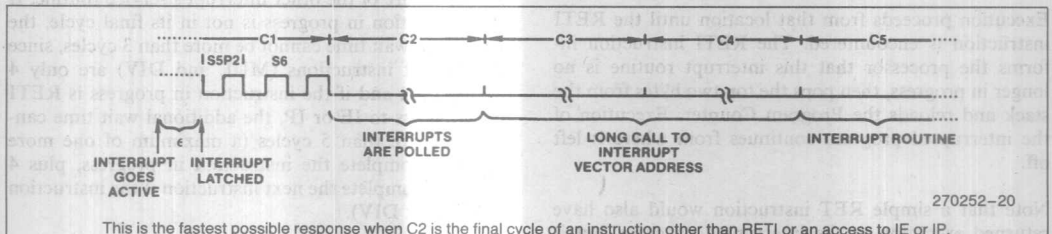
The IP register contains a number of unimplemented bits. IP.7 and IP.6 are vacant in the 8052s, and in the 8051s these and IP.5 are vacant. User software should not write 1s to these bit positions, since they may be used in future MCS-51 products.

How Interrupts Are Handled

The interrupt flags are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. If one of the flags was in a set condition at S5P2 of the preceding cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority level is already in progress.
2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE or IP registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be



This is the fastest possible response when C2 is the final cycle of an instruction other than RETI or an access to IE or IP.

Figure 24. Interrupt Response Timing Diagram

completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IE or IP, then at least *one more* instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at S5P2 of the previous machine cycle. Note then that if an interrupt flag is active but not being responded to for one of the above conditions, if the flag is not *still* active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The polling cycle/LCALL sequence is illustrated in Figure 24.

Note that if an interrupt of higher priority level goes active prior to S5P2 of the machine cycle labeled C3 in Figure 24, then in accordance with the above rules it will be vectored to during C5 and C6, without any instruction of the lower priority routine having been executed.

Thus the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine. In some cases it also clears the flag that generated the interrupt, and in other cases it doesn't. It never clears the Serial Port or Timer 2 flags. This has to be done in the user's software. It clears an external interrupt flag (IE0 or IE1) only if it was transition-activated. The hardware-generated LCALL pushes the contents of the Program Counter onto the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to, as shown below.

Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H
TF2 + EXF2	002BH

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stack and reloads the Program Counter. Execution of the interrupted program continues from where it left off.

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking an interrupt was still in progress.

External Interrupts

The external sources can be programmed to be level-activated or transition-activated by setting or clearing bit IT1 or IT0 in Register TCON. If ITx = 0, external interrupt x is triggered by a detected low at the $\overline{\text{INTx}}$ pin. If ITx = 1, external interrupt x is edge-triggered. In this mode if successive samples of the $\overline{\text{INTx}}$ pin show a high in one cycle and a low in the next cycle, interrupt request flag IEx in TCON is set. Flag bit IEx then requests the interrupt.

Since the external interrupt pins are sampled once each machine cycle, an input high or low should hold for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one cycle, and then hold it low for at least one cycle to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.

If the external interrupt is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

Response Time

The $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ levels are inverted and latched into IE0 and IE1 at S5P2 of every machine cycle. The values are not actually polled by the circuitry until the next machine cycle. If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine. Figure 24 shows interrupt response timings.

A longer response time would result if the request is blocked by one of the 3 previously listed conditions. If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is not in its final cycle, the additional wait time cannot be more than 3 cycles, since the longest instructions (MUL and DIV) are only 4 cycles long, and if the instruction in progress is RETI or an access to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV).

Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 9 cycles.

SINGLE-STEP OPERATION

The 8051 interrupt structure allows single-step execution with very little software overhead. As previously noted, an interrupt request will not be responded to while an interrupt of equal priority level is still in progress, nor will it be responded to after RETI until at least one other instruction has been executed. Thus, once an interrupt routine has been entered, it cannot be re-entered until at least one instruction of the interrupted program is executed. One way to use this feature for single-stop operation is to program one of the external interrupts (say, $\overline{INT0}$) to be level-activated. The service routine for the interrupt will terminate with the following code:

```
JNB   P3.2,$  ;Wait Here Till INT0 Goes High
JB    P3.2,$  ;Now Wait Here Till it Goes Low
RETI          :Go Back and Execute One Instruction
```

Now if the $\overline{INT0}$ pin, which is also the P3.2 pin, is held normally low, the CPU will go right into the External Interrupt 0 routine and stay there until $\overline{INT0}$ is pulsed (from low to high to low). Then it will execute RETI, go back to the task program, execute one instruction, and immediately re-enter the External Interrupt 0 routine to await the next pulsing of P3.2. One step of the task program is executed each time P3.2 is pulsed.

RESET

The reset input is the RST pin, which is the input to a Schmitt Trigger.

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), *while the oscillator is running.* The CPU responds by generating an internal reset, with the timing shown in Figure 25.

The external reset signal is asynchronous to the internal clock. The RST pin is sampled during State 5 Phase 2 of every machine cycle. The port pins will maintain their current activities for 19 oscillator periods after a logic 1 has been sampled at the RST pin; that is, for 19 to 31 oscillator periods after the external reset signal has been applied to the RST pin.

While the RST pin is high, ALE and PSEN are weakly pulled high. After RST is pulled low, it will take 1 to 2 machine cycles for ALE and PSEN to start clocking. For this reason, other devices can not be synchronized to the internal timings of the 8051.

The internal reset algorithm writes 0s to all the SFRs except the port latches, the Stack Pointer, and SBUF. The port latches are initialized to FFH, the Stack Pointer to 07H, and SBUF is indeterminate. Table 3 lists the SFRs and their reset values.

The internal RAM is not affected by reset. On power up the RAM content is indeterminate.

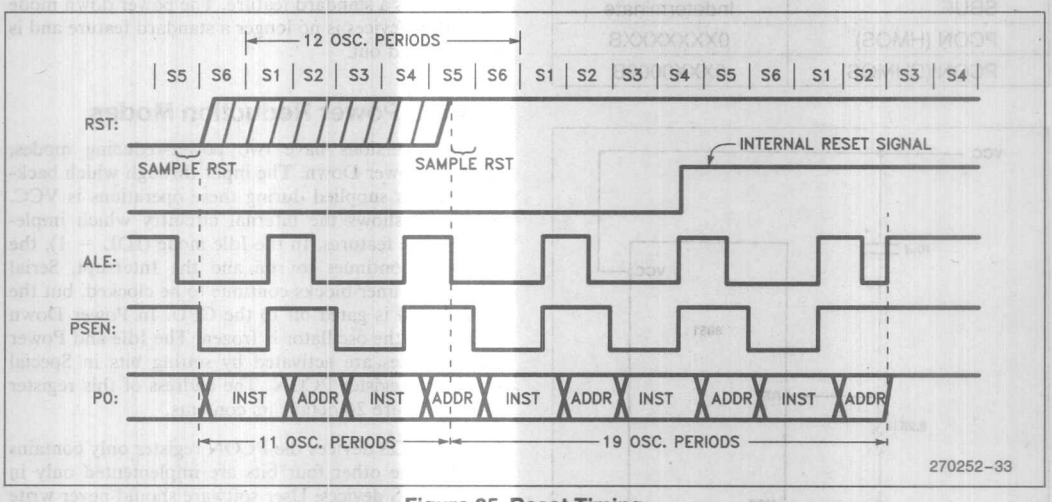


Figure 25. Reset Timing

Table 3. Reset Values of the SFRs

SFR Name	Reset Value
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0–P3	FFH
IP (8051)	XXX00000B
IP (8052)	XX000000B
IE (8051)	0XX00000B
IE (8052)	0X000000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
TH2 (8052)	00H
TL2 (8052)	00H
RCAP2H (8052)	00H
RCAP2L (8052)	00H
SCON	00H
SBUF	Indeterminate
PCON (HMOS)	0XXXXXXXB
PCON (CHMOS)	0XXX0000B

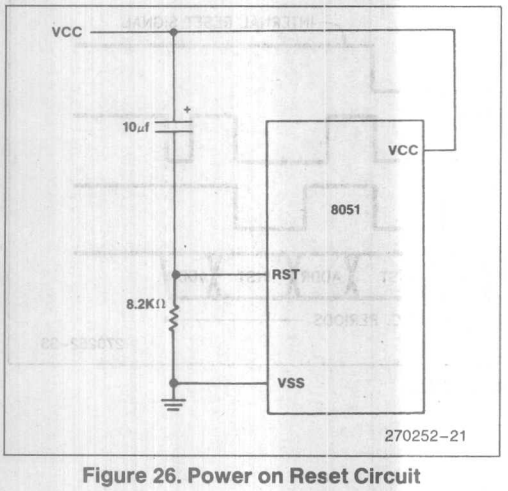


Figure 26. Power on Reset Circuit

POWER-ON RESET

An automatic reset can be obtained when VCC is turned on by connecting the RST pin to VCC through a 10 μf capacitor and to VSS through an 8.2 KΩ resistor, providing the VCC risetime does not exceed a millisecond and the oscillator start-up time does not exceed 10 milliseconds. This power-on reset circuit is shown in Figure 26. The CHMOS devices do not require the 8.2K pulldown resistor, although its presence does no harm.

When power is turned on the circuit holds the RST pin high for an amount of time that depends on the value of the capacitor and the rate at which it charges. To ensure a good reset the RST pin must be high long enough to allow the oscillator time to start up (normally a few msec) plus two machine cycles.

Note that the port pins will be in a random state until the oscillator has started and the internal reset algorithm has written 1s to them.

With this circuit, reducing VCC quickly to 0 causes the RST pin voltage to momentarily fall below 0V. However, this voltage is internally limited, and will not harm the device.

POWER-SAVING MODES OF OPERATION

For applications where power consumption is critical the CHMOS version provides power reduced modes of operation as a standard feature. The power down mode in HMOS devices is no longer a standard feature and is being phased out.

CHMOS Power Reduction Modes

CHMOS versions have two power-reducing modes, Idle and Power Down. The input through which back-up power is supplied during these operations is VCC. Figure 27 shows the internal circuitry which implements these features. In the Idle mode (IDL = 1), the oscillator continues to run and the Interrupt, Serial Port, and Timer blocks continue to be clocked, but the clock signal is gated off to the CPU. In Power Down (PD = 1), the oscillator is frozen. The Idle and Power Down modes are activated by setting bits in Special Function Register PCON. The address of this register is 87H. Figure 26 details its contents.

In the HMOS devices the PCON register only contains SMOD. The other four bits are implemented only in the CHMOS devices. User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

IDLE MODE

An instruction that sets PCON.0 causes that to be the last instruction executed before going into the Idle

mode. In the Idle mode, the internal clock signal is gated off to the CPU, but not to the Interrupt, Timer, and Serial Port functions. The CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical states they had at the time Idle was activated. ALE and $\overline{PSEN}$ hold at logic high levels.

There are two ways to terminate the Idle. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that put the device into Idle.

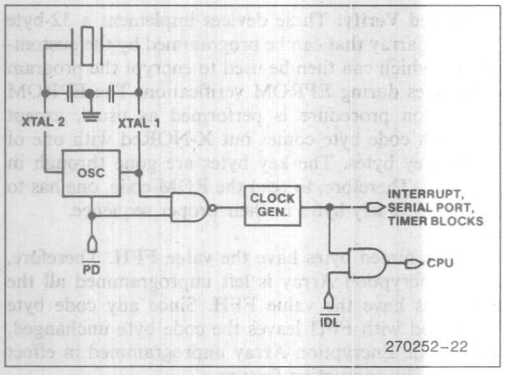


Figure 27. Idle and Power Down Hardware

(MSB)							(LSB)
SMOD	-	-	-	GF1	GF0	PD	IDL

Symbol	Position	Name and Function
SMOD	PCON.7	Double Baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the Serial Port is used in modes 1, 2, or 3.
—	PCON.6	(Reserved)
—	PCON.5	(Reserved)
—	PCON.4	(Reserved)
GF1	PCON.3	General-purpose flag bit.
GF0	PCON.2	General-purpose flag bit.
PD	PCON.1	Power Down bit. Setting this bit activates power down operation.
IDL	PCON.0	Idle mode bit. Setting this bit activates idle mode operation.

If 1s are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XXX0000).
In the HMOS devices the PCON register only contains SMOD. The other four bits are implemented only in the CHMOS devices. User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

Figure 28. PCON: Power Control Register

The flag bits GF0 and GF1 can be used to give an indication if an interrupt occurred during normal operation or during an Idle. For example, an instruction that activates Idle can also set one or both flag bits. When Idle is terminated by an interrupt, the interrupt service routine can examine the flag bits.

The other way of terminating the Idle mode is with a hardware reset. Since the clock oscillator is still running, the hardware reset needs to be held active for only two machine cycles (24 oscillator periods) to complete the reset.

The signal at the RST pin clears the IDL bit directly and asynchronously. At this time the CPU resumes program execution from where it left off; that is, at the instruction following the one that invoked the Idle Mode. As shown in Figure 25, two or three machine cycles of program execution may take place before the internal reset algorithm takes control. On-chip hardware inhibits access to the internal RAM during this time, but access to the port pins is not inhibited. To eliminate the possibility of unexpected outputs at the port pins, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external Data RAM.

POWER DOWN MODE

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the Power Down mode. In the Power Down mode, the on-chip oscillator is stopped. With the clock frozen, all functions are stopped, but the on-chip RAM and Special Function Registers are held. The port pins output the values held by their respective SFRs. ALE and $\overline{PSEN}$ output lows.

The only exit from Power Down for the 80C51 is a hardware reset. Reset redefines all the SFRs, but does not change the on-chip RAM.

In the Power Down mode of operation, VCC can be reduced to as low as 2V. Care must be taken, however, to ensure that VCC is not reduced before the Power Down mode is invoked, and that VCC is restored to its normal operating level, before the Power Down mode is terminated. The reset that terminates Power Down also frees the oscillator. The reset should not be activated before VCC is restored to its normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize (normally less than 10 msec).

EPROM VERSIONS

The EPROM versions of these devices are listed in Table 4. The 8751H programs at VPP = 21V using one 50 msec $\overline{PROG}$ pulse per byte programmed. This results in a total programming time (4K bytes) of approximately 4 minutes.

Table 4. EPROM Versions of the 8051 and 8052

Device Name	EPROM Version	EPROM Bytes	Ckt Type	VPP	Time Required to Program Entire Array
8051	(8751)	4K	HMOS	21.0V	4 minutes
8051AH	8751H	4K	HMOS	21.0V	4 minutes
80C51BH	87C51	4K	CHMOS	12.75V	13 seconds
8052AH	8752BH	8K	HMOS	12.75V	26 seconds

The 8752BH and 87C51 use the faster "Quick-Pulse" programming™ algorithm. These devices program at $\overline{VPP}$ = 12.75V using a series of twenty-five 100 μs $\overline{PROG}$ pulses per byte programmed. This results in a total programming time of approximately 26 seconds for the 8752BH (8K bytes) and 13 seconds for the 87C51 (4K bytes).

Detailed procedures for programming and verifying each device are given in the data sheets.

EXPOSURE TO LIGHT

It is good practice to cover the EPROM window with an opaque label when the device is in operation. This is not so much to protect the EPROM array from inadvertent erasure, but to protect the RAM and other on-chip logic. Allowing light to impinge on the silicon die while the device is operating can cause logical malfunction.

Program Memory Locks

In some microcontroller applications it is desirable that the Program Memory be secure from software piracy. Intel has responded to this need by implementing a Program Memory locking scheme in some of the MCS-51 devices. While it is impossible for anyone to guarantee absolute security against all levels of technological sophistication, the Program Memory locks in the MCS-51 devices will present a formidable barrier against illegal readout of protected software.

8751H

The 8751H contains a lock bit which, once programmed, denies electrical access by any external means to the on-chip Program Memory. The effect of this lock bit is that while it is programmed the internal Program Memory can not be read out, the device can not be further programmed, and it *can not execute external Program Memory*. Erasing the EPROM array deactivates the lock bit and restores the device's full functionality. It can then be re-programmed.

The procedure for programming the lock bit is detailed in the 8751H data sheet.

87C51 AND 8752BH

The 87C51 and 8752BH contain two Program Memory locking schemes: Encrypted Verify and Lock Bits.

Encrypted Verify: These devices implement a 32-byte EPROM array that can be programmed by the customer, and which can then be used to encrypt the program code bytes during EPROM verification. The EPROM verification procedure is performed as usual, except that each code byte comes out X-NORed with one of the 32 key bytes. The key bytes are gone through in sequence. Therefore, to read the ROM code, one has to know the 32 key bytes in their proper sequence.

Unprogrammed bytes have the value FFH. Therefore, if the Encryption Array is left unprogrammed all the key bytes have the value FFH. Since any code byte X-NORed with FFH leaves the code byte unchanged, leaving the Encryption Array unprogrammed in effect bypasses the encryption feature.

Lock Bits: Also on the chip are two Lock Bits which can be left unprogrammed (U) or programmed (P) to obtain the following features:

Bit 2	Bit 1	Additional Features
U	U	None
U	P	• Externally fetched code can not access internal Program Memory. • Further programming disabled.
P	U	(Reserved for Future definition.)
P	P	• Externally fetched code can not access internal Program Memory. • Further programming disabled. • Program verification is disabled.

When Lock Bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

ONCE Mode in the 87C51

The ONCE ("on-circuit emulation") mode facilitates testing and debugging of systems using the 87C51 without the 87C51 having to be removed from the circuit. The ONCE mode is invoked by:

1. Pull ALE low while the device is in reset and $\overline{PSEN}$ is high;
2. Hold ALE low as RST is deactivated.

While the device is in ONCE mode, the Port 0 pins go into a float state, and the other port pins and ALE and $\overline{PSEN}$ are weakly pulled high. The oscillator circuit remains active. While the 87C51 is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored after a normal reset is applied.

THE ON-CHIP OSCILLATORS

HMOS Versions

The on-chip oscillator circuitry for the HMOS (HMOS-I and HMOS-II) members of the MCS-51 family is a single stage linear inverter (Figure 29), intended for use as a crystal-controlled, positive reactance oscillator (Figure 30). In this application the crystal is operated in its fundamental response mode as an inductive reactance in parallel resonance with capacitance external to the crystal.

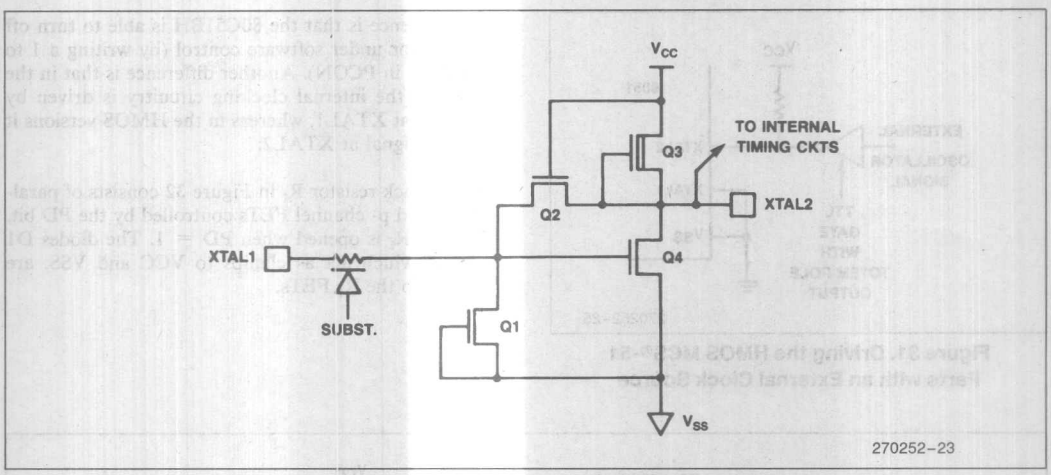


Figure 29. On-Chip Oscillator Circuitry in the HMOS Versions of the MCS®-51 Family

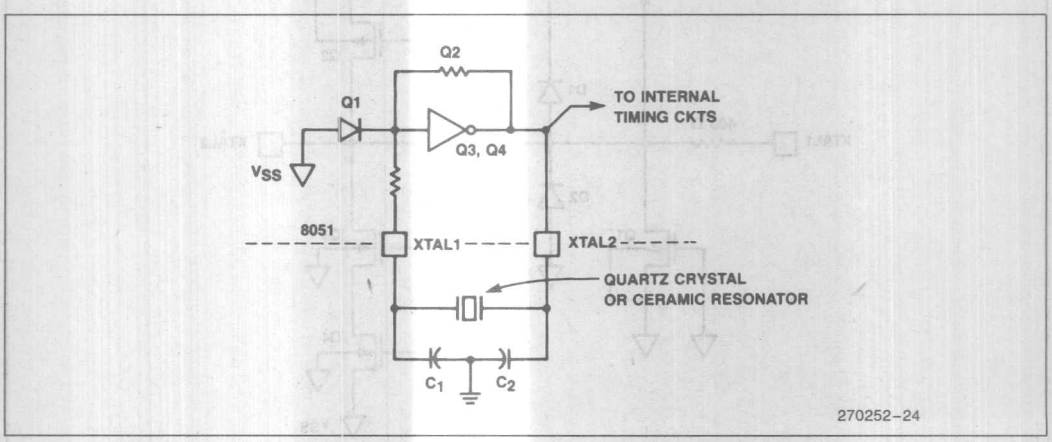


Figure 30. Using the HMOS On-Chip Oscillator

The crystal specifications and capacitance values (C1 and C2 in Figure 30) are not critical. 30 pF can be used in these positions at any frequency with good quality crystals. A ceramic resonator can be used in place of the crystal in cost-sensitive applications. When a ceramic resonator is used, C1 and C2 are normally selected to be of somewhat higher values, typically, 47 pF. The manufacturer of the ceramic resonator should be consulted for recommendations on the values of these capacitors.

A more in-depth discussion of crystal specifications, ceramic resonators, and the selection of values for C1 and C2 can be found in Application Note AP-155, "Oscillators for Microcontrollers," which is included in this manual.

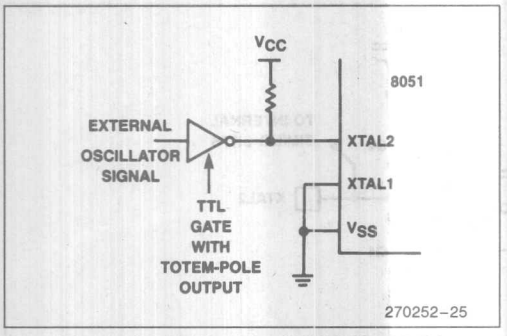


Figure 31. Driving the HMOS MCS®-51 Parts with an External Clock Source

To drive the HMOS parts with an external clock source, apply the external clock signal to XTAL2, and ground XTAL1, as shown in Figure 31. A pullup resistor may be used (to increase noise margin), but is optional if VOH of the driving gate exceeds the VIH MIN specification of XTAL2.

CHMOS VERSIONS

The on-chip oscillator circuitry for the 80C51BH, shown in Figure 32, consists of a single stage linear inverter intended for use as a crystal-controlled, positive reactance oscillator in the same manner as the HMOS parts. However, there are some important differences.

One difference is that the 80C51BH is able to turn off its oscillator under software control (by writing a 1 to the PD bit in PCON). Another difference is that in the 80C51BH the internal clocking circuitry is driven by the signal at XTAL1, whereas in the HMOS versions it is by the signal at XTAL2.

The feedback resistor R_f in Figure 32 consists of paralleled n- and p- channel FETs controlled by the PD bit, such that R_f is opened when PD = 1. The diodes D1 and D2, which act as clamps to VCC and VSS, are parasitic to the R_f FETs.

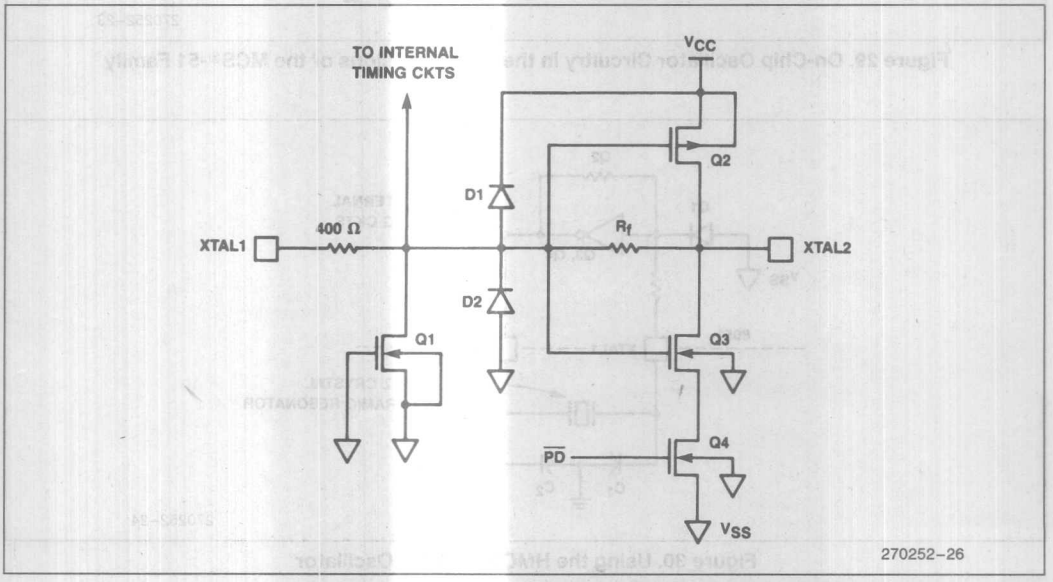


Figure 32. On-Chip Oscillator Circuitry in the CHMOS Versions of the MCS®-51 Family

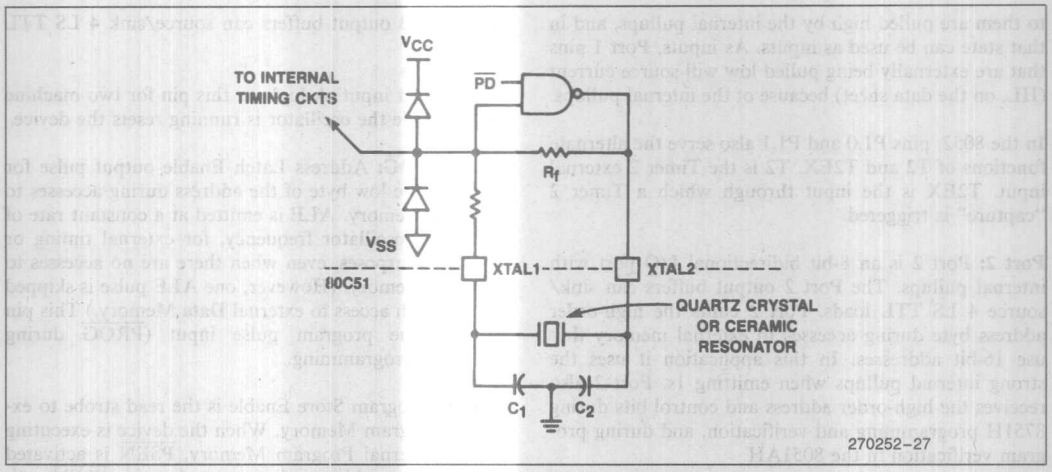


Figure 33. Using the CHMOS On-Chip Oscillator

The oscillator can be used with the same external components as the HMOS versions, as shown in Figure 33. Typically, C1 = C2 = 30 pF when the feedback element is a quartz crystal, and C1 = C2 = 47 pF when a ceramic resonator is used.

To drive the CHMOS parts with an external clock source, apply the external clock signal to XTAL1, and leave XTAL2 float, as shown in Figure 34.

The reason for this change from the way the HMOS part is driven can be seen by comparing Figures 29 and 32. In the HMOS devices the internal timing circuits are driven by the signal at XTAL2. In the CHMOS devices the internal timing circuits are driven by the signal at XTAL1.

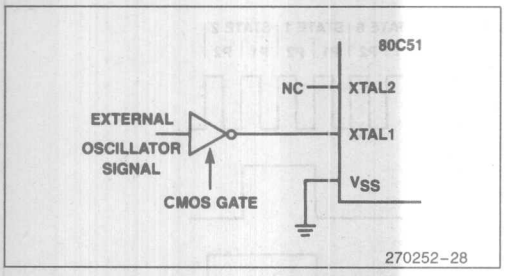


Figure 34. Driving the CHMOS MCS®-51 Parts with an External Clock Source

INTERNAL TIMING

Figures 35 through 38 show when the various strobe and port signals are clocked internally. The figures do not show rise and fall times of the signals, nor do they show propagation delays between the XTAL2 signal and events at other pins.

Rise and fall times are dependent on the external loading that each pin must drive. They are often taken to be something in the neighborhood of 10 nsec, measured between 0.8V and 2.0V.

Propagation delays are different for different pins. For a given pin they vary with pin loading, temperature, VCC, and manufacturing lot. If the XTAL2 waveform is taken as the timing reference, prop delays may vary from 25 to 125 nsec.

The AC Timings section of the data sheets do not reference any timing to the XTAL2 waveform. Rather, they relate the critical edges of control and input signals to each other. The timings published in the data sheets include the effects of propagation delays under the specified test conditions.

MCS®-51 PIN DESCRIPTIONS

VCC: Supply voltage.

VSS: Circuit ground potential.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an open drain output port it can sink 8 LS TTL loads. Port 0 pins that have 1s written to them float, and in that state will function as high-impedance imputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external memory. In this application it uses strong internal pullups when emitting 1s. Port 0 also emits code bytes during program verification. In that application, external pullups are required.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL loads. Port 1 pins that have 1s written

to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL, on the data sheet) because of the internal pullups.

In the 8052, pins P1.0 and P1.1 also serve the alternate functions of T2 and T2EX. T2 is the Timer 2 external input. T2EX is the input through which a Timer 2 "capture" is triggered.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL loads. Port 2 emits the high-order address byte during accesses to external memory that use 16-bit addresses. In this application it uses the strong internal pullups when emitting 1s. Port 2 also receives the high-order address and control bits during 8751H programming and verification, and during program verification in the 8051AH.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. It also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

The Port 3 output buffers can source/sink 4 LS TTL loads.

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. ALE is emitted at a constant rate of $\frac{1}{6}$ of the oscillator frequency, for external timing or clocking purposes, even when there are no accesses to external memory. (However, one ALE pulse is skipped during each access to external **Data** Memory.) This pin is also the program pulse input ($\overline{PROG}$ during EPROM programming.

$\overline{PSEN}$: Program Store Enable is the read strobe to external Program Memory. When the device is executing out of external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle (except that two $\overline{PSEN}$ activations are skipped during accesses to external **Data** Memory). $\overline{PSEN}$ is not activated when the device is executing out of internal Program Memory.

$\overline{EA}$/VPP: When $\overline{EA}$ is held high the CPU executes out of internal Program Memory (unless the Program Counter exceeds 0FFFH in the 8051AH, or 1FFFH in the 8052). Holding $\overline{EA}$ low forces the CPU to execute out of external memory regardless of the Program Counter value. In the 8031AH and 8032, $\overline{EA}$ must be extremely wired low. In the EPROM devices, this pin also receives the programming supply voltage (VPP) during EPROM programming.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

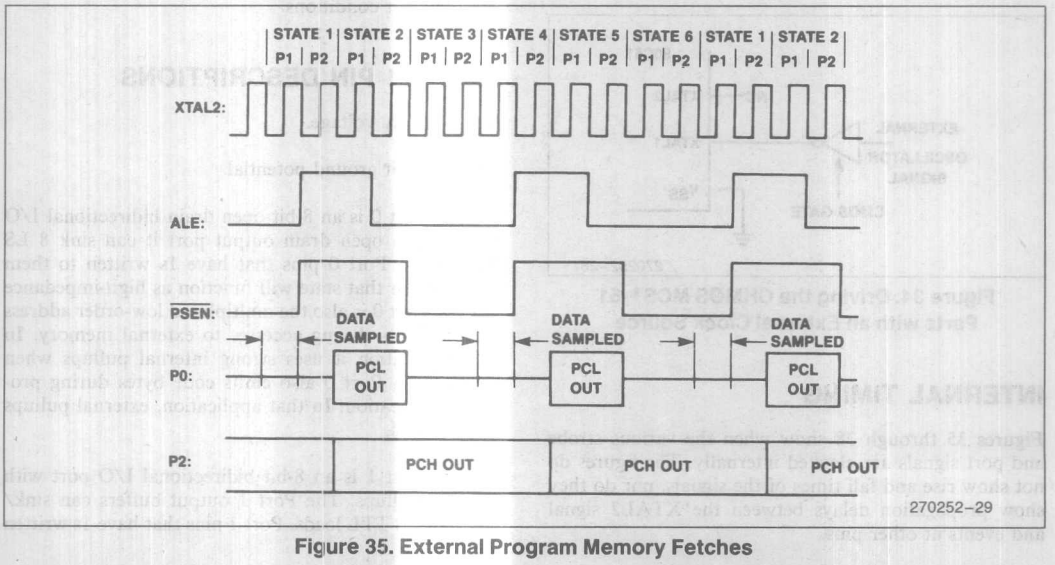


Figure 35. External Program Memory Fetches

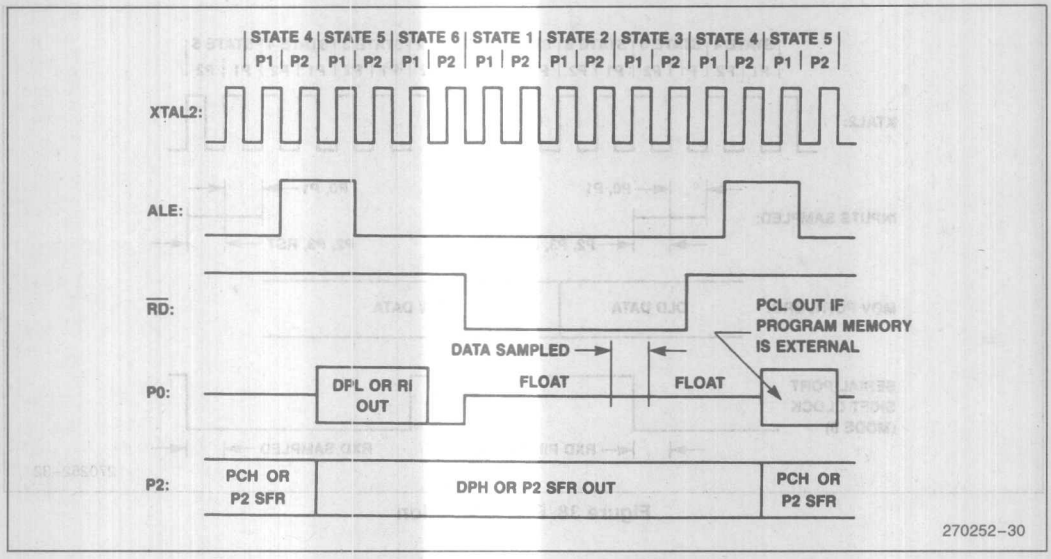


Figure 36. External Data Memory Read Cycle

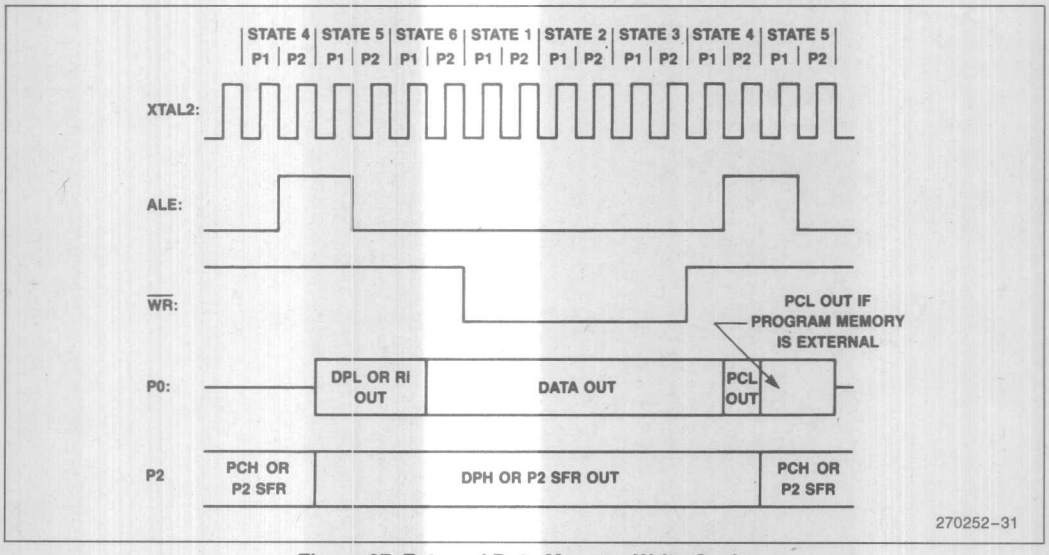


Figure 37. External Data Memory Write Cycle

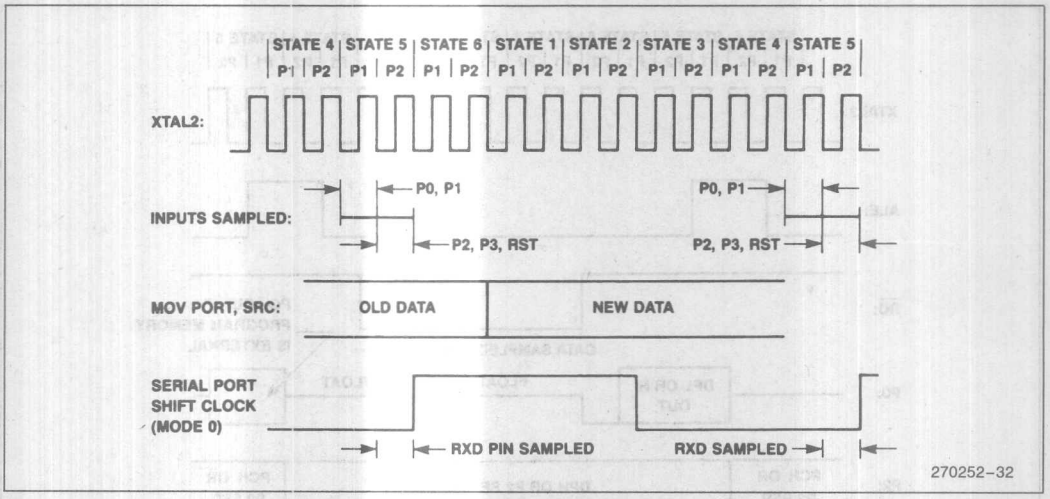


Figure 38. Port Operation

Hardware Description of the 83C51FA

Hardware Description of the 83C51FA

7

HARDWARE DESCRIPTION OF THE 83C51FA (83C252)

INTRODUCTION

The 83C51FA is an 8-bit control-oriented microcontroller based on the 8051 architecture. The 83C51FA is an enhanced version of the 80C51BH and incorporates many new features. These features include:

- Programmable Counter Array with
  - Compare/Capture
  - High Speed Output
  - Pulse Width Modulator
  - Watchdog Timer
- Programmable Serial Channel
  - Automatic Address Recognition
  - Framing Error Detection
- Enhanced Power Down Mode
- 16-Bit Up/Down Timer/Counter
- 8K Factory Mask ROM
- 256 Bytes of On-Chip Data RAM
- 7 Interrupt Sources

The 83C51FA uses the standard 8051 instruction set and is pin for pin compatible with the existing MCS®-51 products. However, the numbering system for the 83C51FA is slightly different. The 83C51FA is the factory masked ROM device; the 80C51FA is the ROMless device; and the 87C51FA is the EPROM device.

It is assumed that the reader is familiar with the 8051 architecture. For more detailed information on the 8051, consult the "Hardware Description of the 8051 and 8052" chapter.

OVERVIEW OF THE PCA

The Programmable Timer/Counter Array (PCA) consists of a 16-bit counter and five 16-bit compare/capture modules. Each compare/capture module has its own mode register, CCAPMn, which is used to configure the module. The compare/capture modules and the PCA counter share Port 1 pins for hardware interfacing as shown below:

Port Pin	Name	Function
P1.2	ECI	External Count Input to the PCA
P1.3	CEX0	External I/O for Compare/Capture Module 0
P1.4	CEX1	External I/O for Compare/Capture Module 1
P1.5	CEX2	External I/O for Compare/Capture Module 2
P1.6	CEX3	External I/O for Compare/Capture Module 3
P1.7	CEX4	External I/O for Compare/Capture Module 4

The time-base for the PCA is a programmable 16-bit timer/counter. This timer is the only one that can serve the PCA. This timer is started or stopped by setting or clearing bit CR in the Special Function Register CCON, and can be programmed to count any of the following signals (where Fosc is the 83C51FA oscillator frequency):

- Fosc/12
  The Counter increments once per machine cycle.
- Fosc/4
  With a 16 MHz crystal, the counter increments once every 250 ns.
- Timer 0 overflow
  The counter is incremented whenever Timer 0 overflows. This mode allows a programmable input frequency to the PCA.
- External input on ECI pin
  The counter is incremented when a 1-to-0 transition is detected on the ECI pin. The counter is limited to input frequencies of Fosc/8 in this mode.

The 16-bit PCA timer/counter can also be programmed to either run or pause when the CPU is in Idle mode.

Each of the five 16-bit compare/capture modules can be programmed to do one of the following:

- 16-bit capture, positive edge activated.
- 16-bit capture, negative edge activated.
- 16-bit capture, both positive and negative edge activated.
- 16-bit software timer.
- High-speed output.
- 8-bit Pulse Width Modulator (PWM).

In addition, Compare/Capture module 4 can be used as a Watchdog Timer.

When any of the compare/capture modules are programmed to the capture mode or the 16-bit Timer/ High speed output mode, an interrupt can be generated when the module executes its function.

DESCRIPTION OF THE PCA HARDWARE

The time base for the PCA is a 16-bit timer/counter consisting of registers CH and CL (high and low bytes of the count value), controlled by Special Function Register CCON (Figure 1) and a mode register CMOD (Figure 2).

CCON contains bits CF (Counter Flag) which gets set by hardware when the counter rolls over and CR (Counter Run) which is used to turn the counter on and off. It also contains interrupt flags from each of the five PCA modules.

CF	CR	—	CCF4	CCF3	CCF2	CCF1	CCF0

Address = 0D8H Reset Value = 00X00000B

Symbol	Position	Function
CF	CCON.7	PCA Counter Overflow flag. Set by hardware when the counter rolls over. CF flags an interrupt if bit ECF in CMOD is set. CF may be set by either hardware or software. It can only be cleared by software.
CR	CCON.6	Counter Run control bit. Set by software to turn the PCA counter on. Clear by software to turn the PCA counter off.
—	CCON.5	Not implemented, reserved for future use.*
CCF4	CCON.4	PCA Module 4 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF3	CCON.3	PCA Module 3 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF2	CCON.2	PCA Module 2 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF1	CCON.1	PCA Module 1 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF0	CCON.0	PCA Module 0 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.

Figure 1. CCON: Counter Control Register

NOTE:
*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

CMOD contains the following bits:

CIDL— selects whether the PCA counter continues to run in Idle Mode

WDTE— enables the Watchdog Timer function

CPS1 and CPS0—select the counter input

ECF— enables CF to generate an interrupt.

CIDL	WDTE	—	—	—	CPS1	CPS0	ECF

Address = 0D9H Reset Value = 00XXX000B

Symbol	Position	Function
CIDL	CMOD.7	Counter Idle control: CIDL = 0 programs PCA Counter to continue functioning during Idle mode. CIDL = 1 programs it to be gated off during Idle.
WDTE	CMOD.6	Watchdog Timer Enable: WDTE = 0 disables Watchdog Timer function. WDTE = 1 enables it.
—	CMOD.5	Not implemented, reserved for future use.*
—	CMOD.4	Not implemented, reserved for future use.*
—	CMOD.3	Not implemented, reserved for future use.*
CPS1	CMOD.2	Count Pulse Select bit 1.
CPS0	CMOD.1	Count Pulse Select bit 0.
		CPS1 / CPS0 / PCA Count Pulse Selected 0 0 Internal clock, Fosc/12 0 1 Internal clock, Fosc/4 1 0 Timer 0 overflow 1 1 External clock at ECI pin (P1.2) (maximum rate = Fosc/8)
ECF	CMOD.0	Enable Counter Overflow interrupt: ECF = 1 enables CF bit in CCON to generate an interrupt. ECF = 0 disables that function of CF.

Figure 2. CMOD: Counter Mode Register

NOTE:
*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

Each of the five PCA modules has a Compare/Capture Mode register, CCAPMn, n = 0 through 4 (Figure 3). The following bits in each CCAPMn register define that module's function.

ECOMn— enables that module's comparator function

CAPPn— enables the capture function on positive transitions at the CEXn pin

CAPNn— enables the capture function on negative transitions at the CEXn pin

MATn— enables a comparator match to set the corresponding CCFn flag

TOGn— enables a comparator match to toggle the CEXn pin

PWMn— enables the PWM output at the CEXn pin

ECCFn— enables any Compare/Capture event to flag an interrupt

—	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn

Addresses = 0DAH (n=0)
0DBH (n=1)
0DCH (n=2)
0DDH (n=3)
0DEH (n=4)

Reset Value = X0000000B

Symbol	Position	Function
—	CCAPMn.7	Not implemented, reserved for future use.*
ECOMn	CCAPMn.6	ECOMn = 1 enables the comparator function. This bit is automatically cleared by any write to the CCAPnL register, and automatically set by any write to the CCAPnH register. This prevents unintended matches from occurring during writes to the 16-bit Compare/Capture register.
CAPPn	CCAPMn.5	Positive edge capture enable. When CAPPn = 1, a positive transition at the CEXn pin triggers a 16-bit capture from the PCA counter to this module's Compare/Capture register.
CAPNn	CCAPMn.4	Negative edge capture enable. When CAPNn = 1, a negative transition at the CEXn pin triggers a 16-bit capture from the PCA counter to this module's Compare/Capture register.
MATn	CCAPMn.3	When MATn = 1, a match of the PCA Counter with this module's Compare/Capture register causes the CCFn bit in CCON to be set, flagging an interrupt.
TOGn	CCAPMn.2	When TOGn = 1, a match of the PCA Counter with this module's Compare/Capture register causes the CEXn pin to toggle.
PWMn	CCAPMn.1	When PWMn = 1, CEXn is driven high when the low byte of the PCA Counter (CL) matches the low byte of this module's Compare/Capture register (CCAPnL). When CL rolls over to 00H, the CEXn pin is driven low and CCAPnL is updated with the value in CCAPnH. This enables the CEXn pin to be used as a pulse width modulated output. Software varies the pulse width by writing to CCAPnH.
ECCFn	CCAPMn.0	Enables Compare/Capture Flag CCFn in the CCON register to generate an interrupt.

Figure 3. CCAPMn: Compare/Capture Mode Register for PCA Module n

NOTE:
*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

There are 6 modes of operation for each of the 5 PCA modules. Shown below are the combinations of bits in the CCAPMn register that are valid and have a defined function. Invalid combinations will produce undefined results.

ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	Module Function
0	0	0	0	0	0	0	No operation
X	1	0	0	0	0	X	16-bit capture by a positive-edge trigger on CEXn
X	0	1	0	0	0	X	16-bit capture by a negative-edge trigger on CEXn
X	1	1	0	0	0	X	16-bit capture by a transition on CEXn
1	0	0	1	0	0	X	16-bit software timer
1	0	0	1	1	0	X	High Speed Output
1	0	0	0	0	1	0	8-bit PWM

X = Don't Care

16-Bit Timer/Counter

The 5 Compare/Capture modules share a 16-bit timer/counter as their "time base". The timer/counter, shown in Figure 4, can be programmed to increment in 4 different ways. The modes are shown below with the setup values in the CMOD register:

CPS1	CPS0	Method of Incrementing
0	0	Internally clocked at Oscillator Frequency /12
0	1	Internally clocked at Oscillator Frequency /4
1	0	Incremented when Timer 0 overflows
1	1	Externally clocked on Pin P1.2/ECI. (Limited to Oscillator Frequency /8)

16-Bit Capture Mode

Setting CAPPn and/or CAPNn puts Compare/Capture module n into Input Capture mode, as shown in Figure 5. The external input pins CEX0 through CEX4 are sampled for a transition. When a valid transition is detected for the current mode of operation (rising edge, falling edge, or either edge), CL is transferred into the CCAPnL register, and CH is transferred into CCAPnH. The resulting value in the Capture Registers, CCAPnL and CCAPnH, reflect the values in CL and CH at the time a transition was detected on the CEXn pin. The event flags an interrupt if bit ECCFn is set.

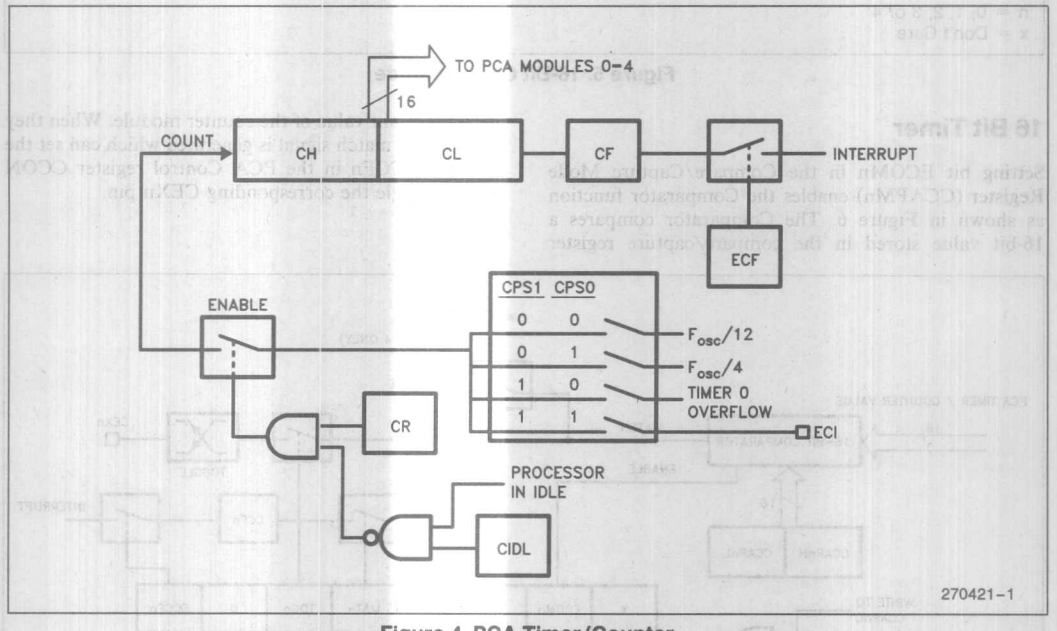


Figure 4. PCA Timer/Counter

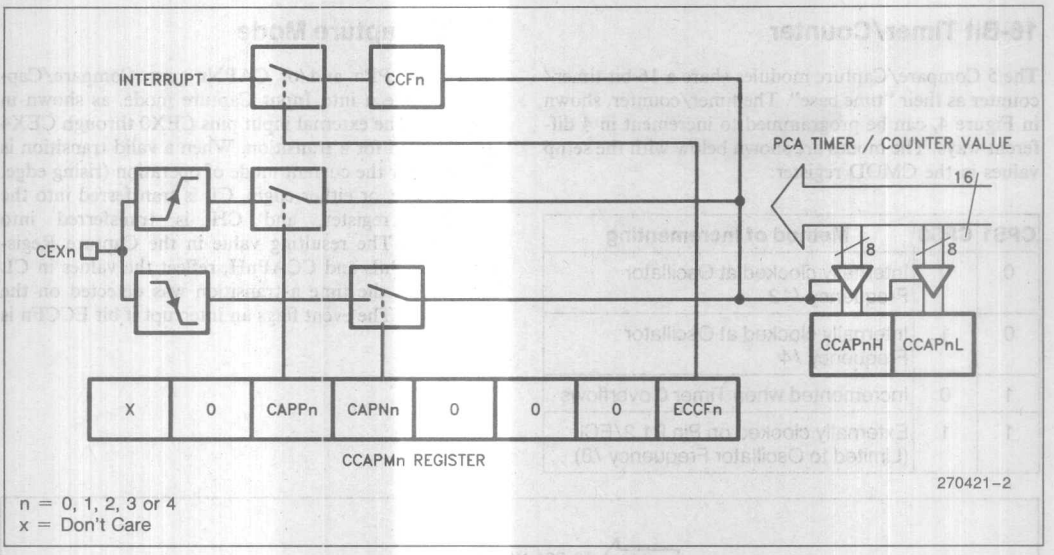


Figure 5. 16-Bit Capture Mode

16 Bit Timer

Setting bit ECOMn in the Compare/Capture Mode Register (CCAPMn) enables the Comparator function as shown in Figure 6. The Comparator compares a 16-bit value stored in the compare/capture register with the count value of the counter module. When they are equal, a match signal is generated which can set the status bit CCFn in the PCA Control register CCON and/or toggle the corresponding CEXn pin.

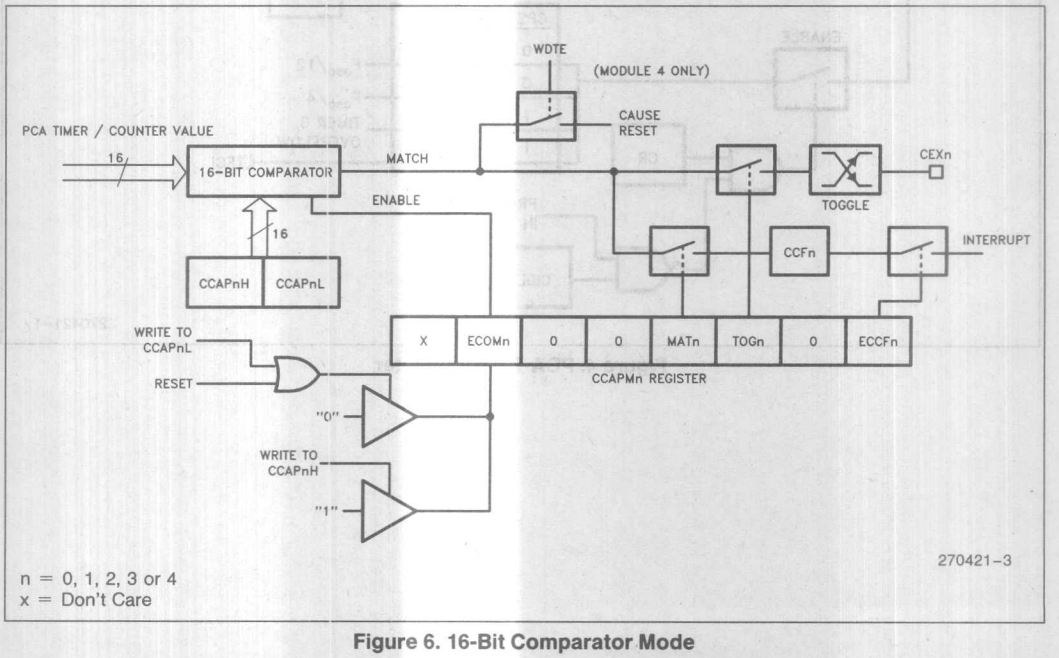


Figure 6. 16-Bit Comparator Mode

Bit ECOMn is set by software and is initially cleared during reset. It also gets cleared when a write to CCAPnL register happens and is set if CCAPnH is written to. This feature prevents action until the complete 16-bit value is loaded into the CCAPnH/L register if the low value is written to the 16-bit register first.

When the MATn (Match) bit is set in the Compare/Capture Mode Register, the corresponding module in the PCA is configured as a 16-bit timer. When the value in the 16-bit Compare/Capture register is equal to the 16-bit value on the Count Bus, the hardware sets the CCFn flag. This bit flags an interrupt if ECCFn is also set.

High Speed Output

When programmed as a timer, the PCA module, during every cycle, compares the contents of the 16-bit timer with the preset value of its Compare registers. When a match occurs, if bit TOGn is set, the module reverses the logic level of its I/O pin, and/or can generate an interrupt as shown in Figure 6. When the PCA module is configured in this manner as a High Speed Output, the user, by setting or clearing the pin in software, can select whether the module's output pin will change from a logical 0 to a logical 1 or vice versa.

Pulse Width Modulator Mode

Any or all of the 5 modules of the PCA can be programmed to be a Pulse Width Modulator as shown in Figure 7. In this mode, the PWM output can be used to convert digital data to an analog signal by simple external circuitry. The frequency of the PWM depends on which of the four clock sources is selected for the PCA Timer. With a 16 MHz crystal the maximum frequency of the output waveform of the PWM is 15.6 KHz. The duty cycle of the waveform is controlled by the contents of an 8-bit register (CCAPnH) that can be programmed to be any integer from 0 to 255.

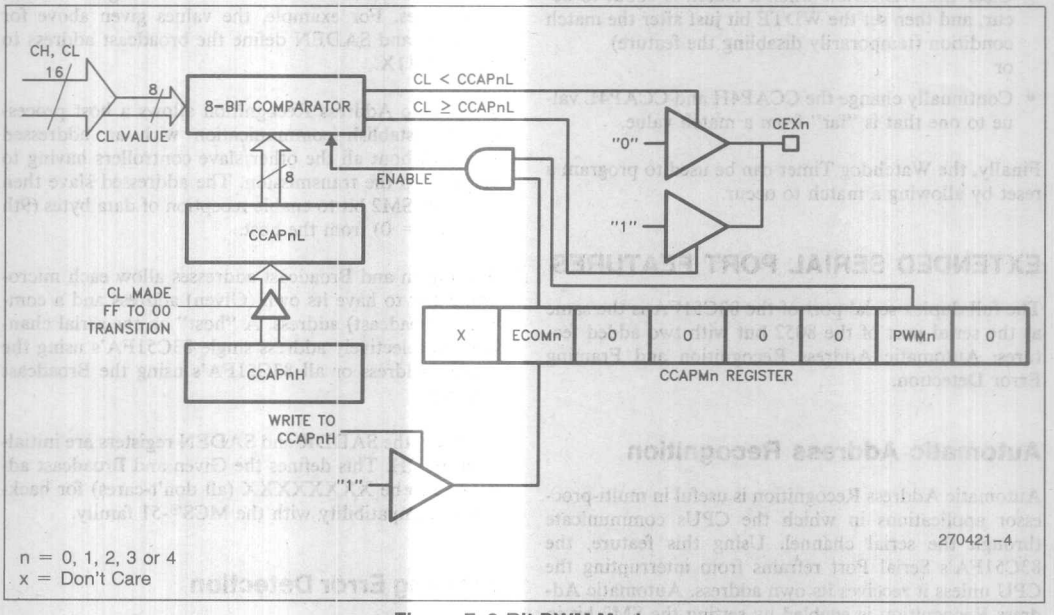


Figure 7. 8-Bit PWM Mode

Watch Dog Timer Mode

A Watchdog Timer is a circuit that automatically invokes a reset unless the system being watched sends regular hold-off signals to the Watchdog. These circuits are used in applications that are subject to electrical noise, power glitches, electrostatic discharges, etc., or where high reliability is required with hands-off operation.

In this mode, every time the count in the PCA counter module matches the value stored in compare/capture module 4, an internal reset is generated. The bit that selects this mode is WDTE in the CMOD register. Compare/capture module 4 should be set up to be a 16-bit timer or a High Speed Output in the Watchdog Timer mode.

To hold off the reset, the user's software can:

- Continually reset the PCA 16-bit timer value to a lower value than the reset value in module 4.
- Clear the WDTE bit when a match is about to occur, and then set the WDTE bit just after the match condition (temporarily disabling the feature).
  or
- Continually change the CCAP4H and CCAP4L value to one that is "far" from a match value.

Finally, the Watchdog Timer can be used to program a reset by allowing a match to occur.

EXTENDED SERIAL PORT FEATURES

The full duplex serial port of the 83C51FA is the same as the serial port of the 8052 but with two added features: Automatic Address Recognition and Framing Error Detection.

Automatic Address Recognition

Automatic Address Recognition is useful in multi-processor applications in which the CPUs communicate through the serial channel. Using this feature, the 83C51FA's Serial Port refrains from interrupting the CPU unless it receives its own address. Automatic Address Recognition is enabled by setting the SM2 bit in SCON.

Normally the Serial Port would be configured into either of the 9-bit modes (modes 2 and 3). In these modes, if SM2 is set, the Receive Interrupt flag RI is not activated unless the received byte is an address byte (9th data bit = 1), and the address corresponds to either a Given Address or a Broadcast Address.

The feature works the same way in the 8-bit mode (mode 1) as in the 9-bit modes, except that the stop bit takes the place of the 9th data bit. That is, if SM2 is set, RI is not activated unless the received byte agrees with either the Given or Broadcast address and is terminated by a valid stop bit.

The Given Address is specified by the contents of two new SFRs: SADDR and SADEN. The 83C51FA's individual address is defined in SADDR. SADEN is a mask byte that defines don't-cares in SADDR to form the Given Address. For example,

SADDR = 01010110
SADEN = 11111100

specify the Given Address to be 010101XX.

The Broadcast Address is formed from the logical OR of SADDR and SADEN. Zeros in the logical OR are don't-cares. For example, the values given above for SADDR and SADEN define the broadcast address to be 1111111X.

Automatic Address Recognition allows a host processor to establish communication with an addressed slave, without all the other slave controllers having to respond to the transmission. The addressed slave then clears its SM2 bit to enable reception of data bytes (9th data bit = 0) from the host.

The Given and Broadcast addresses allow each microcontroller to have its own (Given) address and a common (Broadcast) address. A "host" on the serial channel can selectively address single 83C51FA's using the Given Address or all 83C51FA's using the Broadcast Address.

On reset, the SADDR and SADEN registers are initialized to 00H. This defines the Given and Broadcast addresses to be XXXXXXXX (all don't-cares) for backwards compatibility with the MCS®-51 family.

Framing Error Detection

Another new feature of the Serial Port is Framing Error Detection. This allows the receiving controller to check the stop bit in modes 1, 2, or 3. A missing stop bit causes a Framing Error bit, FE, to be set. The FE bit can then be checked in software immediately after each reception to detect the lack of a valid stop bit. A missing stop bit can be caused, for example, by noise on the serial lines, or by two CPUs trying to transmit at the same time.

The FE bit, once set, must be cleared in software. A valid stop bit does not cause the FE bit to be cleared.

The FE bit resides in SCON, and has the same bit address as the SM0 bit. A new control bit in the PCON register determines if accesses to the SM0/FE bit address are to SM0 or to FE. The new control bit in PCON is called SMOD0, and resides at PCON.6 (Figure 8). If SMOD0 = 0, then accesses to SCON.7 are to SM0. If SMOD0 = 1, then accesses to SCON.7 are to FE.

REDUCED POWER MODES

Idle Mode

Idle Mode is the same in the 83C51FA as in the 80C51BH. Note that the PCA can be programmed to either pause or continue operations during Idle.

Power Down Mode

The Power Down Mode on the 83C51FA differs from the 80C51BH in one respect: the 83C51FA can exit Power Down with either a hardware Reset or an External Interrupt. (The 80C51BH can only exit Power Down with a hardware Reset.) An exit with an External Interrupt allows not only the on-chip RAM to be saved but also the Special Function Registers.

The External Interrupt, INT0 or INT1, must be enabled and configured as level-sensitive to properly terminate Power Down. Also the interrupt should not be executed before V_{CC} is restored to its normal operating level, and must be held down long enough for the oscillator to restart and stabilize. Once the interrupt is serviced, the next instruction executed after RETI will be the one following the instruction that put the device in Power Down.

Power Off Flag

A Power Off Flag, POF, has been added to the PCON register (Figure 8). This flag is set by hardware when VCC comes up, and can be set or cleared by software. This allows one to distinguish between a "cold start" reset and a "warm start" reset.

A cold start reset is one that is coincident with VCC being turned on to the device after it was turned off. A warm start reset is one that occurs after the device has already been powered up and running. A warm start reset could be generated, for example, by a Watchdog Timer, or as an exit from Power Down Mode.

To use the feature, one checks the POF bit in software immediately after reset. POF = 1 would indicate a cold start. The software then clears POF, and commences its tasks. POF = 0 immediately after reset would indicate a warm start.

VCC must remain above 3 volts for POF to retain a 0.

TIMER 2 AS AN UP/DOWN COUNTER

Timer 2 is a general purpose 16-bit timer/counter which is present in the 8052 and in the 83C51FA. Timer 2 has the same functionality in both of these devices except that in the 8052 Timer 2 can only count up, and in the 83C51FA Timer 2 can be programmed to count up or down. The option to count up or down becomes available when the Timer is configured to its 16-bit auto-reload mode.

SMOD1	SMOD0	X	POF	GF1	GF0	PD	IDL

Address = 087H Reset Value = 00XX0000B

POF Power Off Flag. Set by hardware on the rising edge of V_{CC}. Set or cleared by software. This flag allows detection of a power failure caused reset. V_{CC} must remain above 3V to retain this bit.

SMOD0 When set, Read/Write accesses to SCON.7 are to the FE bit. When clear, Read/Write accesses to SCON.7 are to the SM0 bit.

SMOD1 Same as the SMOD bit in the MSC-51 architecture. The additional bits are defined to be compatible with the 8052 and 80C51BH.

Figure 8. PCON: Power Control Register

X	X	X	X	X	X	X	DCEN

Address = 0C9H Reset Value = XXXX XXX0B

DCEN When set, this bit allows Timer 2 to be configured as an up/down counter.

Figure 9. T2MOD: Timer 2 Mode Control Register

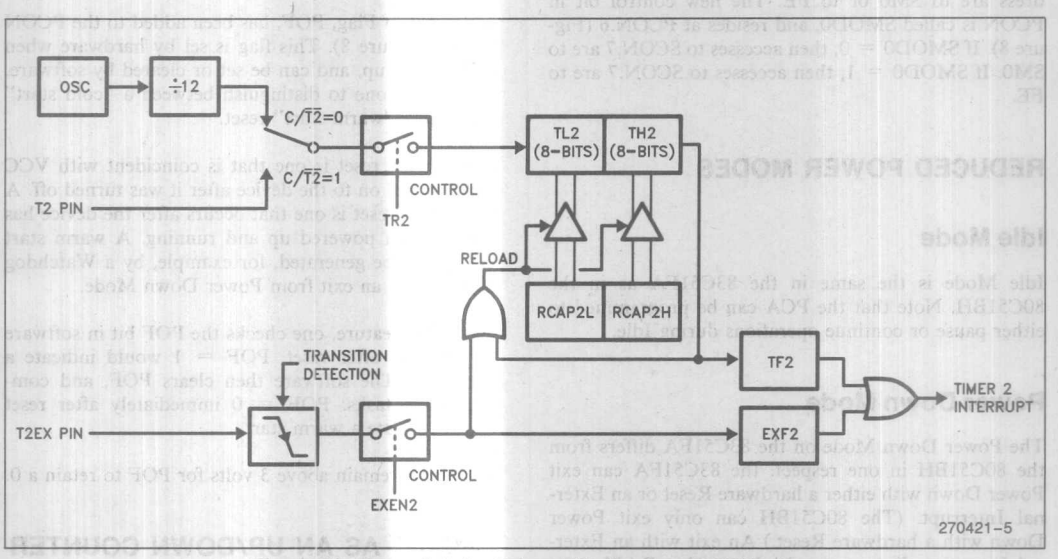


Figure 10. Timer 2 Auto-Reload Mode when DCEN = 0

The Special Function Register T2MOD (present in the 83C51FA but not in the 8052) contains a bit named DCEN (Down Counter Enable). T2MOD is shown in Figure 9. When this bit is clear (0), the Timer 2 Auto-Reload Mode in the 83C51FA is exactly the same as in the 8052. Figure 10 shows Timer 2 in Auto-Reload Mode with DCEN = 0.

When DCEN is set (1), the Timer 2 Auto-Reload Mode takes the form shown in Figure 11. The T2EX pin now controls the direction of count. A logic 1 at T2EX makes Timer 2 count up. A logic 0 at T2EX makes Timer 2 count down. Also, the EXF2 bit toggles every time Timer 2 overflows or underflows. In this operating mode, the EXF2 bit does not flag an interrupt.

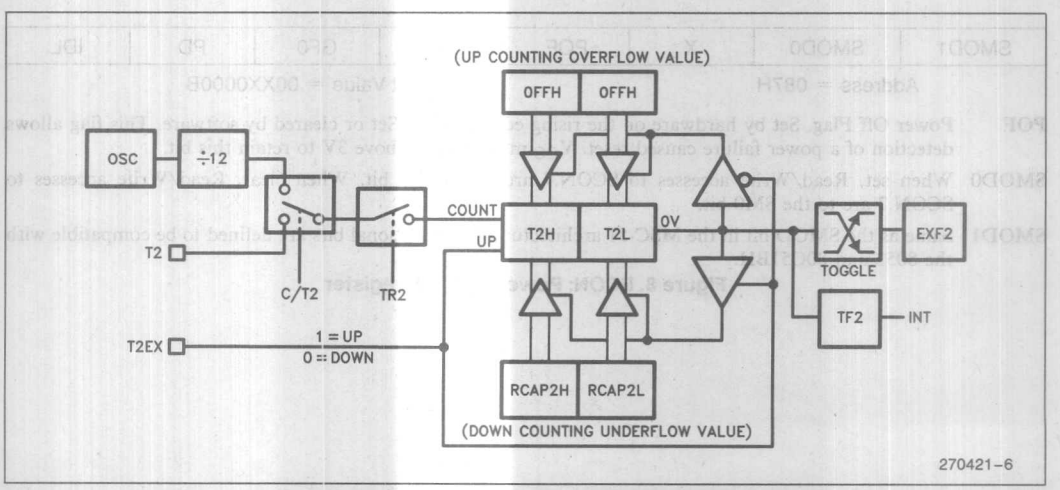


Figure 11. Timer 2 Auto-Reload Mode when DCEN = 1

UPPER 128 BYTES OF RAM

The 83C51FA implements a full 256 bytes of on-chip data RAM. As in the 8052, the upper 128 bytes occupy a parallel address space to the Special Function Registers. That means they have the same addresses, but they are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the CPU knows whether the access is to the upper 128 bytes of data RAM or to SFR space by the addressing mode used in the instruction. Instructions that use direct addressing access SFR space. For example,

MOV 0A0H, #data

accesses the SFR at location 0A0H (which is P2). Instructions that use indirect addressing access the upper 128 bytes of data RAM. For example,

MOV @R0, #data

where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H).

Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

PIN DESCRIPTION

The 83C51FA is Pin-for-Pin compatible with the 80C51BH. Port 1 on the 83C51FA has 8052 functionality and additionally serves the PCA as shown below.

Port 1 pins and Alternate Functions.

Port Pin	Name	Function
P1.0	T2	External Count Input to Timer 2
P1.1	T2EX	Timer 2 Capture/Reload Trigger
P1.2	ECI	External Count Input to the PCA
P1.3	CEX0	External I/O for Compare/Capture Module 0
P1.4	CEX1	External I/O for Compare/Capture Module 1
P1.5	CEX2	External I/O for Compare/Capture Module 2
P1.6	CEX3	External I/O for Compare/Capture Module 3
P1.7	CEX4	External I/O for Compare/Capture Module 4

FUNCTIONS OF PORT 1 PINS

P1.0/T2 may be used as an external count input to Timer 2.

P1.1/T2EX can be used to trigger a capture if Timer 2 is in the Capture Mode, or to trigger a reload if Timer 2 is in the Auto-Reload Mode and DCEN is set to 0. T2EX can also control the count direction if Timer 2 is in the Auto-Reload Mode and DCEN set to 1. Finally, T2EX can be used as an external interrupt if Timer 2 is being used as a baud-rate generator.

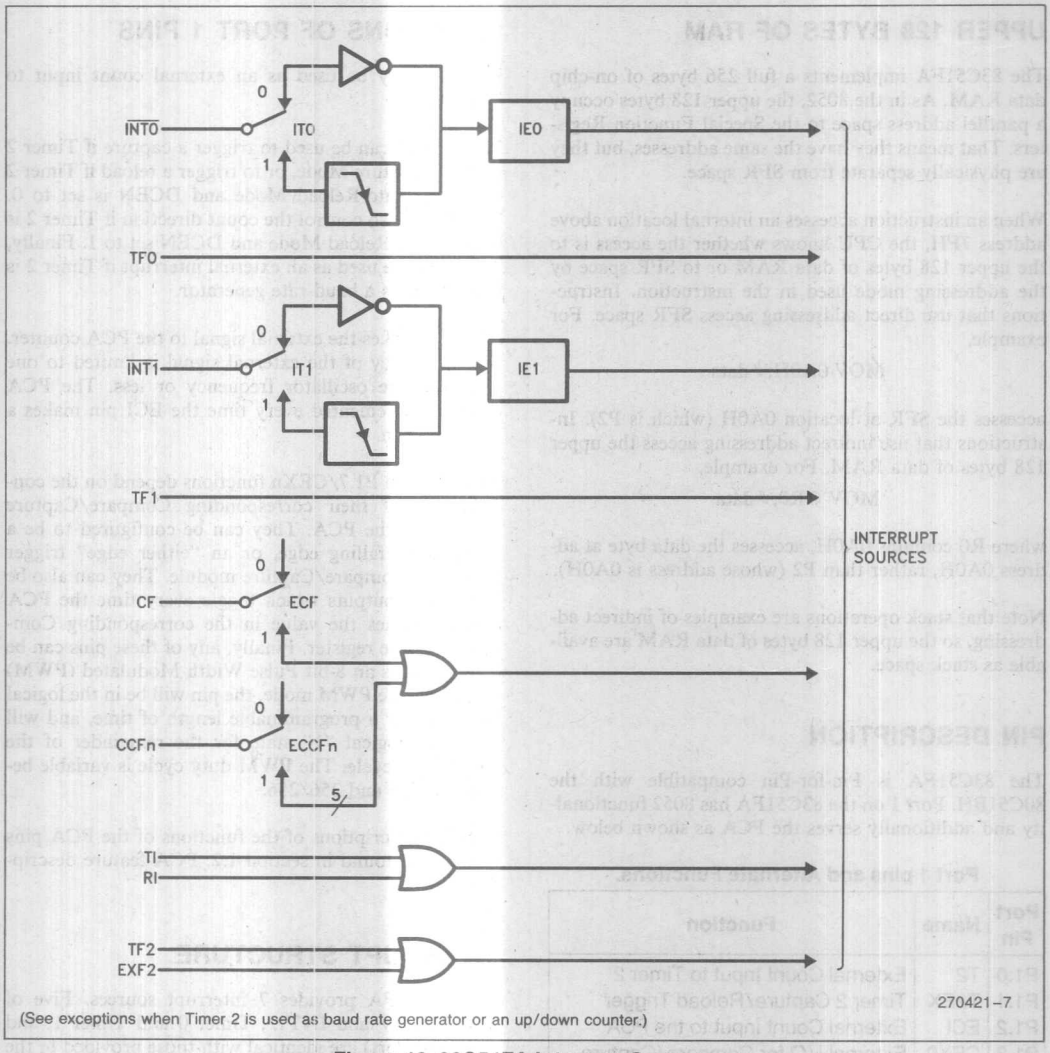
P1.2/ECI takes the external signal to the PCA counter. The frequency of the external signal is limited to one eighth of the oscillator frequency or less. The PCA count is incremented every time the ECI pin makes a 1-0 transition.

P1.3 through P1.7/CEXn functions depend on the configuration of their corresponding Compare/Capture modules in the PCA. They can be configured to be a rising edge, falling edge, or an "either edge" trigger input to a Compare/Capture module. They can also be high speed outputs which toggle every time the PCA count matches the value in the corresponding Compare/Capture register. Finally, any of these pins can be configured as an 8-bit Pulse Width Modulated (PWM) output. In the PWM mode, the pin will be in the logical "0" state for a programmable length of time, and will be in the logical "1" state for the remainder of the PWM duty cycle. The PWM duty cycle is variable between 1/256 and 256/256.

Detailed descriptions of the functions of the PCA pins can also be found in section 1.2, PCA feature description.

INTERRUPT STRUCTURE

The 83C51FA provides 7 interrupt sources. Five of them (INT0/ and INT1/, Timer 0 and Timer 1, and the Serial Port) are identical with those provided in the 80C51BH. The 83C51FA also provides a Timer 2 interrupt which is identical with the Timer 2 interrupt in the 8052, and a PCA interrupt which is only found in the 83C51FA. These interrupt sources are shown in Figure 12.



(See exceptions when Timer 2 is used as baud rate generator or an up/down counter.)

Figure 12. 83C51FA Interrupt Sources

The Timer 2 interrupt is generated by the logical OR of TF2 and EXF2. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared in software.

The PCA interrupt is generated by the logical OR of CF, CCF0, CCF1, CCF2, CCF3, and CCF4. None of these flags is cleared by hardware when the service routine is vectored to. In fact, normally the service routine will have to determine which bit flagged the interrupt and clear that bit in software.

All of the bits that generate interrupts can be set or cleared in software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled in software.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE (Figure 13). Note that IE also contains a global disable bit, EA. If EA is set (1), the interrupts are individually enabled or disabled by their corresponding bits in IE. If EA is clear (0), all interrupts are disabled.

EA	EC	ET2	ES	ET1	EX1	ET0	EX0

Address = 0A8H Reset Value = 00000000B

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, all interrupts are disabled. If EA = 1, each interrupt can be individually enabled or disabled by setting or clearing its enable bit.
EC	IE.6	Enables or disables the PCA interrupt. EC = 1 enables it. EC = 0 disables it.
ET2	IE.5	Enables or disables the Timer 2 interrupt. ET2 = 1 enables it. ET2 = 0 disables it.
ES	IE.4	Enables or disables the Serial Port interrupt. ES = 1 enables it. ES = 0 disables it.
ET1	IE.3	Enables or disables the Timer 1 interrupt. ET1 = 1 enables it. ET1 = 0 disables it.
EX1	IE.2	Enables or disables External Interrupt 1. EX1 = 1 enables it. EX1 = 0 disables it.
ET0	IE.1	Enables or disables the Timer 0 interrupt. ET0 = 1 enables it. ET0 = 0 disables it.
EX0	IE.0	Enables or disables External Interrupt 0. EX0 = 1 enables it. EX0 = 0 disables it.

Figure 13. IE: Interrupt Enable Register

PRIORITY LEVEL STRUCTURE

Each interrupt source can be individually programmed to one of two priority levels by setting or clearing a bit in Special Function Register IP (Figure 14). A low-priority interrupt can be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

—	PPC	PT2	PS	PT1	PX1	PT0	PX0

Address = 0B8H Reset Value = X0000000B

Symbol	Position	Function
—	IP.7	Not implemented, reserved for future use.*
PPC	IP.6	Defines the PCA interrupt priority level. PPC = 1 programs it to the high priority level.
PT2	IP.5	Defines the Timer 2 interrupt priority level. PT2 = 1 programs it to the high priority level.
PS	IP.4	Defines the Serial Port interrupt priority level. PS = 1 programs it to the high priority level.
PT1	IP.3	Defines the Timer 1 interrupt priority level. PT1 = 1 programs it to the high priority level.
PX1	IP.2	Defines the External Interrupt 1 priority level. PX1 = 1 programs it to the high priority level.
PT0	IP.1	Defines the Timer 0 interrupt priority level. PT0 = 1 programs it to the high priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level. PX0 = 1 programs it to the high priority level.

NOTE:
*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

Figure 14. IP: Interrupt Priority Register

If two interrupts of different priority levels are flagged simultaneously, the interrupt request of the higher priority level is serviced. If interrupts of the same priority level are flagged simultaneously, an internal polling sequence determines which interrupt request is serviced. Thus within each priority level there is a second priority structure determined by the following polling sequence:

SOURCE	PRIORITY WITHIN LEVEL
1. IE0	(highest)
2. TF0	
3. IE1	
4. TF1	
5. PCA	
6. RI + TI	
7. TF2 + EXF2	(lowest)

Note that the "priority within level" structure is only used to resolve **simultaneous requests of the same priority level.**

LOCATION OF INTERRUPT SERVICE ROUTINES

The Interrupt Control System acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate service routine. The hardware-generated LCALL pushes the contents of the Program Counter onto the stack (but it does not save the PSW) and reloads the PC with the starting location of the interrupt service routine as shown below.

SOURCE	STARTING ADDRESS OF SERVICE ROUTINE
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H
TF2 + EXF2	002BH
PCA	0033H

Execution proceeds from that location until the RETI instruction is encountered, which terminates the interrupt service routine. Note that the starting addresses of consecutive interrupt service routines are only 8 bytes apart. That means if consecutive interrupts are being used (IE0 and TF0, for example, or TF0 and IE1), and if the first interrupt routine is more than 7 bytes long, then that routine will have to execute a jump out to some other memory location where the service routine can be completed without overlapping the starting address of the next interrupt routine.

Note that, although the polling position of the PCA-generated interrupt is higher than that of the Serial Port, the starting address of the Serial Port interrupt routine is unchanged from the 8051. This is for backwards software compatibility. Similarly, the Timer 2 interrupt starting address is compatible with the 8052. This allows conversion of 8052 (HMOS) designs to the 83C51FA (CHMOS) with no software modification.

SPECIAL FUNCTION REGISTERS

A map of the Special Function Register (SFR) space is shown in Table 1.

Note that not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

User software should not write 1s to these unimplemented locations, since they may be used in future MCS-51 products to invoke new features. In that case the reset or inactive values of the new bits will always be 0, and their active values will be 1.

Table 1. Special Function Register Memory Map and Values After Reset

F8		CH 00000000	CCAP0H XXXXXXXX	CCAP1H XXXXXXXX	CCAP2H XXXXXXXX	CCAP3H XXXXXXXX	CCAP4H XXXXXXXX		FF
F0	* B 00000000								F7
E8		CL 00000000	CCAP0L XXXXXXXX	CCAP1L XXXXXXXX	CCAP2L XXXXXXXX	CCAP3L XXXXXXXX	CCAP4L XXXXXXXX		EF
E0	* ACC 00000000								E7
D8	CCON 00X00000	CMOD 00XXX000	CCAPM0 X0000000	CCAPM1 X0000000	CCAPM2 X0000000	CCAPM3 X0000000	CCAPM4 X0000000		DF
D0	* PSW 00000000								D7
C8	T2CON 00000000	T2MOD XXXXXXX0	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			CF
C0									C7
B8	* IP X0000000	SADEN 00000000							BF
B0	* P3 11111111								B7
A8	* IE 00000000	SADDR 00000000							AF
A0	* P2 11111111								A7
98	* SCON 00000000	* SBUF XXXXXXXX							9F
90	* P1 11111111								97
88	* TCON 00000000	* TMOD 00000000	* TL0 00000000	* TL1 00000000	* TH0 00000000	* TH1 00000000			8F
80	* P0 11111111	* SP 00000111	* DPL 00000000	* DPH 00000000				*PCON ** 00XX0000	87

* = Found in the 8051 core (See 8051 Hardware Description for explanations of these SFRs).
** = See description of PCON SFR. Bit PCON.4 is not affected by reset.
X = Undefined.

SPECIAL FUNCTION REGISTERS

A map of the Special Function Register (SFR) space is shown in Table 1.

Note that not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

Table 1. Special Function Register Memory Map and Values After Reset

F8		CH 00000000	CCAP0H XXXXXXXX	CCAP1H XXXXXXXX	CCAP2H XXXXXXXX	CCAP3H XXXXXXXX	CCAP4H XXXXXXXX	FF
F0	* B 00000000							F7
E8		CL 00000000	CCAP0L XXXXXXXX	CCAP1L XXXXXXXX	CCAP2L XXXXXXXX	CCAP3L XXXXXXXX	CCAP4L XXXXXXXX	EF
E0	* ACC 00000000							E7
D8	CCON 00X00000	CMOD 00XXX000	CCAPM0 X0000000	CCAPM1 X0000000	CCAPM2 X0000000	CCAPM3 X0000000	CCAPM4 X0000000	DF
D0	* PSW 00000000							D7
C8	T2CON 00000000	T2MOD XXXXXXX0	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000		C7
C0								C7
B8	* IP X0000000	SADEN 00000000						BF
B0	* P3 11111111							B7
A8	* IE 00000000	SADDR 00000000						AF
A0	* P2 11111111							A7
98	* SCON 00000000	* SBUF XXXXXXXX						9F
90	* P1 11111111							97
88	* TCON 00000000	* TMOD 00000000	* TL0 00000000	* TL1 00000000	* TH0 00000000	* TH1 00000000		8F
80	* P0 11111111	* SP 00000111	* DPL 00000000	* DPH 00000000			* PCON ** 00XX0000	87

* = Found in the 8051 core (See 8051 Hardware Description)
** = See description of PCON SFR. Bit PCON.4 is not affected by RESET.
X = Undefined.

83C152

8 Hardware Description of the 83C152

HARDWARE DESCRIPTION OF THE 83C152

1.0 INTRODUCTION

The 83C152 Universal Communications Controller is an 8-bit microcontroller designed for the intelligent management of peripheral systems or components. The 83C152 is a derivative of the 80C51BH and retains the same functionality. The 83C152 is fabricated on the same CHMOS III process as the 80C51BH. What makes the 83C152 different is that it has added functions and peripherals to the basic 80C51BH architecture that are supported by new Special Function Registers (SFRs). These enhancements include: a high speed multi-protocol serial communication interface, two channels for DMA transfers, HOLD/HLDA bus control, a fifth I/O port, expanded data memory, and expanded program memory.

In addition to a standard UART, referred to here as Local Serial Channel (LSC), the 83C152 has an onboard multi-protocol communication controller called the Global Serial Channel (GSC). The GSC interface supports SDLC, CSMA/CD, user definable protocols, and a subset of HDLC protocols. The GSC capabilities include: address recognition, collision resolution, CRC generation, flag generation, automatic retransmission, and a hardware based acknowledge feature. This high speed serial channel is capable of implementing the Data Link Layer and the Physical Link Layer as shown in the OSI open systems communication model. This model can be found in the document "Reference Model for Open Systems Interconnection Architecture", ISO/TC97/SC16 N309.

The DMA circuitry consists of two 8-bit DMA channels with 16-bit addressability. The control signals; Read ($\overline{RD}$), Write ($\overline{WR}$), hold and hold acknowledge (HOLD/HLDA) are used to access external memory. The DMA channels are capable of addressing up to 64K bytes (16 bits). The destination or source address can be automatically incremented. The lower 8 bits of the address are multiplexed on the data bus Port 0 and the upper eight bits of address will be on Port 2. Data is transmitted over an 8-bit address/data bus. Up to 64K bytes of data may be transmitted for each DMA activation.

The new I/O port (P4) functions the same as Ports 1-3, found on the 80C51BH.

Internal memory has been doubled in the 83C152. Data memory has been expanded to 256 bytes, and internal program memory has been expanded to 8K bytes.

There are also some specific differences between the 83C152 and the 80C51BH. The first is that the numbering system between the 83C152 and the 80C51BH is slightly different. The 83C152 and the 80C51BH are factory masked ROM devices. The 80C152 and the 80C31BH are ROMless devices which require the use of external program memory. The second difference is that RESET is active low in the 83C152 and active high in the 80C51BH. This is very important to designers who may currently be using the 80C51BH and planning to use the 83C152, or are planning on using both devices on the same board. The third difference is that GF0 and GF1, general purpose flags in PCON, have been renamed GFIEN and XRCLK. GFIEN enables idle flags to be generated in SDLC mode, and XRCLK enables the receiver to be externally clocked. All of the previously unused bits are now being used and interrupt vectors have been added to support the new enhancements. Programmers using old code generated for the 80C51BH will have to examine their programs to ensure that new bits are properly loaded, and that the new interrupt vectors will not interfere with their program.

Throughout the rest of this manual the 80C152 and the 83C152 will be referred to generically as the "C152".

The C152 is based on the 80C51BH architecture and utilizes the same 80C51BH instruction set. Figure 1.1 is a block diagram of the C152. Readers are urged to compare this block diagram with the 80C51BH block diagram. There have been no new instructions added. All the new features and peripherals are supported by an extension of the Special Function Registers (SFRs). Very little of the information pertaining specifically to the 80C51BH core will be discussed in this chapter. The detailed information on such functions as: the instruction set, port operation, timer/counters, etc., can be found in the MCS®-51 Architecture chapter in the Intel Embedded Controller Handbook. Knowledge of the 80C51BH is required to fully understand this manual and the operation of the C152. To gain a basic understanding on the operation of the 80C51BH, the reader should familiarize himself with the entire MCS-51 chapter of the Embedded Controller Handbook.

Another source of information that the reader may find helpful is Intel's LAN Components User's Manual, order number 230814. Inside are descriptions of various protocols, application examples, and application notes dealing with different serial communication environments.

2.0 COMPARISON OF 80C152 AND 80C51BH FEATURES

2.1 Memory Space

A good understanding of the memory space and how it is used in the operation of MCS-51 products is essential. All the enhancements on the C152 are implemented by accessing Special Function Registers (SFRs), added data memory, or added program memory.

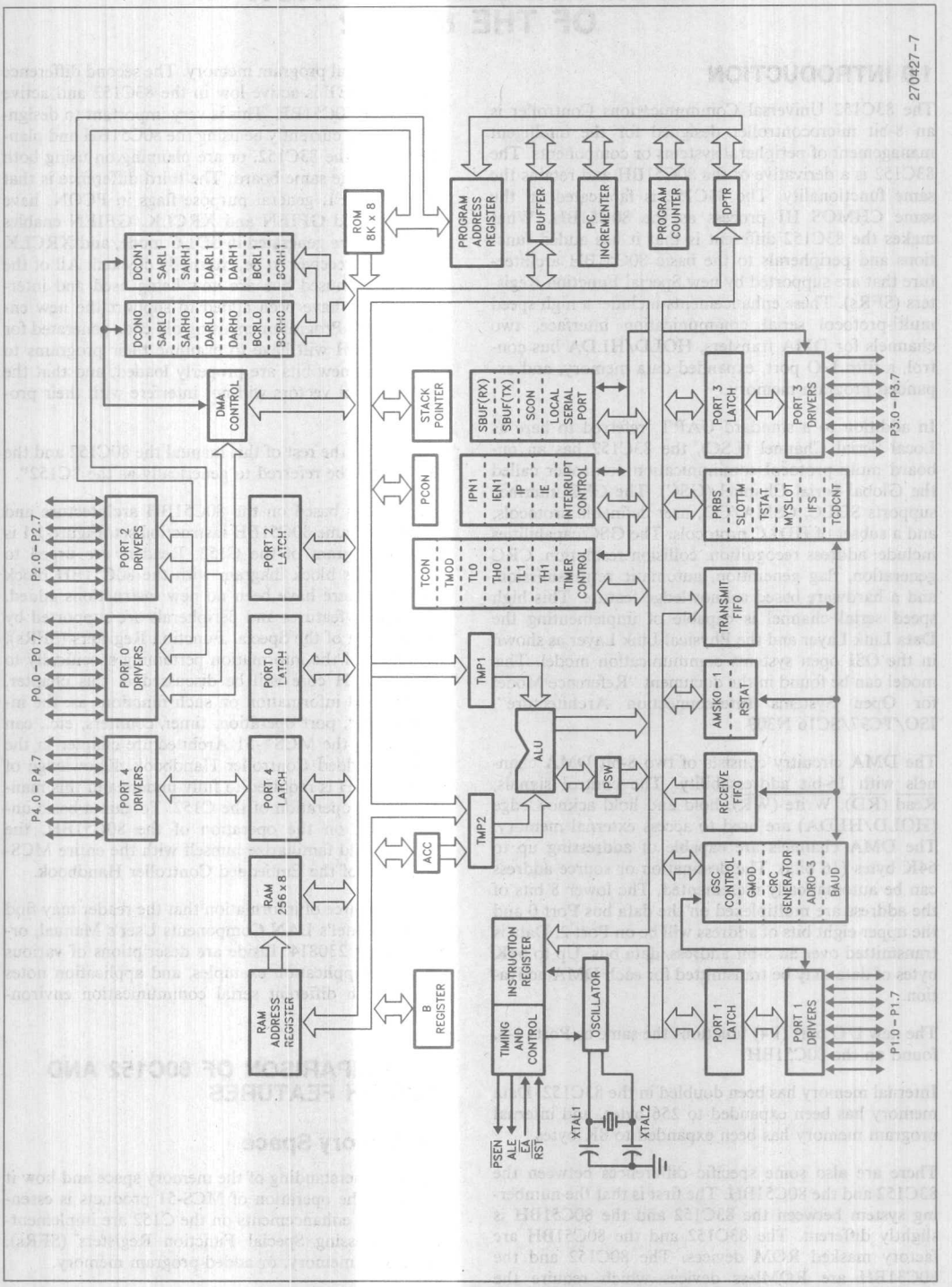


Figure 1.1. Block Diagram

2.1.1 SPECIAL FUNCTION REGISTERS (SFRs)

The following list contains all the SFRs, their names and function. All of the SFRs of the 80C51BH are retained and for a detailed explanation of their operation, please refer to the chapter, "Hardware Description of the 8051 and 8052" that is found in the Embedded Controller Handbook. An overview of the new SFRs is found in Section 2.2.1.1, with a detailed explanation in Section 3.7 and Section 4.5.

2.1.1.1 New SFRs

The following descriptions are quick overviews of the new SFRs, and not intended to give a complete understanding of their use. The reader should refer to the detailed explanation in Section 3 for the GSC SFRs, and Section 4 for the DMA SFRs.

ADR 0,1,2,3 - (95H, 0A5H, 0B5H, 0C5H) Contains the four bytes for address matching during GSC operation.

AMSK0 - (0D5H) Selects "don't care" bits to be used with ADR0.

AMSK1 - (0E5H) Selects "don't care" bits to be used with ADR1.

BAUD - (94H) Contains the programmable value for the baud rate generator for the GSC. The baud rate will equal (fosc)/((BAUD + 1) × 8).

BCRL0 - (0E2H) Contains the low byte of a count-down counter that determines when the DMA access for Channel 0 is complete.

BCRH0 - (0E3H) Contains the high byte for count-down counter for Channel 0.

BCRL1 - (0F2H) Same as BCRL0 except for DMA Channel 1.

BCRH1 - (0F3H) Same as BCRH0 except for DMA Channel 1.

BKOFF - (0C4H) An 8-bit count-down timer used with the CSMA/CD resolution algorithm.

DARL0 - (0C2H) Contains the low byte of the destination address for DMA Channel 0.

DARH0 - (0C3H) Contains the high byte of the destination address for DMA Channel 0.

DARL1 - (0D2H) Same as DARL0 except for DMA Channel 1.

DARH1 - (0D3H) Same as DARH0 except for DMA Channel 1.

DCON0 - (92H) Contains the Destination Address Space bit (DAS), Increment Destination Address bit (IDA), Source Address Space bit (SAS), Increment Source Address bit (ISA), DMA Channel Mode bit (DM), Transfer Mode bit (TM), DMA Done bit (DONE), and the GO bit (GO). DCON0 is used to control DMA Channel 0.

DCON1 - (93H) Same as DCON0 except this is for DMA Channel 1.

GMOD - (84H) Contains the Protocol bit (PR), the Preamble Length (PL1,0), CRC Type (CT), Address Length (AL), Mode select (M1,0), and External Transmit Clock (TXC). This register is used for GSC operation only.

IEN1 - (0C8H) Interrupt enable register for DMA and GSC interrupts.

IFS - (0A4H) Determines the number of bit times separating transmitted frames.

IPN1 - (0F8H) Interrupt priority register for DMA and GSC interrupts.

MYSLOT - (0F5H) Contains the Jamming mode bit (DCJ), the Deterministic Collision Resolution Algorithm bit (DCR), and the DCR slot address for the GSC.

P4 - (0C0H) Contains the memory "image" of Port 4.

PRBS - (0E4H) Contains a pseudo-random number to be used in CSMA/CD backoff algorithms. May be read or written to by user software.

RFIFO - (F4H) RFIFO is used to access a 3-byte FIFO that contains the receive data from the GSC.

RSTAT - (0E8H) Contains the Hardware Based Acknowledge Enable bit (HABEN), Global Receive Enable bit (GREN), Receive FIFO Not Empty bit (RFNE), Receive Done bit (RDN), CRC Error bit (CRCE), Alignment Error bit (AE), Receiver Collision/Abort detect bit (RCABT), and the Overrun bit (OVR), used with both DMA and GSC.

SARL0 - (0A2H) Contains the low byte of the source address for DMA transfers.

SARH0 - (0A3H) Contains the high byte of the source address for DMA transfers.

SARL1 - (0B2H) Same as SARL0 but for DMA Channel 1.

SARH1 - (0B3H) Same as SARH1 but for DMA Channel 1.

SLOTTM - (0B4H) Determines the length of the slot time in CSMA/CD.

TCDCNT - (0D4H) Contains the number of collisions in the current frame if using CSMA/CD GSC.

Old(O)/New(N)	Name	Addr	Function
O	A	0E0H	ACCUMULATOR
N	ADR0	095H	GSC MATCH ADDRESS 0
N	ADR1	0A5H	GSC MATCH ADDRESS 1
N	ADR2	0B5H	GSC MATCH ADDRESS 2
N	ADR3	0C5H	GSC MATCH ADDRESS 3
N	AMSK0	0D5H	GSC ADDRESS MASK 0
N	AMSK1	OE5H	GSC ADDRESS MASK 1
O	B	0F0H	B REGISTER
N	BAUD	094H	GSC BAUD RATE
N	BCRL0	0E2H	DMA BYTE COUNT 0 (LOW)
N	BCRH0	0E3H	DMA BYTE COUNT 0 (HIGH)
N	BCRL1	0F2H	DMA BYTE COUNT 1 (LOW)
N	BCRH1	0F3H	DMA BYTE COUNT 1 (HIGH)
N	BKOFF	0C4H	GSC BACKOFF TIMER
N	DARL0	0C2H	DMA DESTINATION ADDR 0 (LOW)
N	DARH0	0C3H	DMA DESTINATION ADDR 0 (HIGH)
N	DARL1	0D2H	DMA DESTINATION ADDR 1 (LOW)
N	DARH1	0D3H	DMA DESTINATION ADDR 1 (HIGH)
N	DCON0	092H	DMA CONTROL 0
N	DCON1	093H	DMA CONTROL 1
O	DPH	083H	DATA POINTER (HIGH)
O	DPL	082H	DATA POINTER (LOW)
N	GMOD	084H	GSC MODE
O	IE	0A8H	INTERRUPT ENABLE REGISTER 0
N	IEN1	0C8H	INTERRUPT ENABLE REGISTER 1
N	IFS	0A4H	GSC INTERFRAME SPACING
O	IP	0B8H	INTERRUPT PRIORITY REGISTER 0
N	IPN1	0F8H	INTERRUPT PRIORITY REGISTER 1
N	MYSLOT	0F5H	GSC SLOT ADDRESS
O	P0	080H	PORT 0
O	P1	090H	PORT 1
O	P2	0A0H	PORT 2
O	P3	0B0H	PORT 3
N	P4	0C0H	PORT 4
O	PCON	087H	POWER CONTROL
N	PRBS	0E4H	GSC PSEUDO-RANDOM SEQUENCE
O	PSW	0D0H	PROGRAM STATUS WORD
N	RFIFO	0F4H	GSC RECEIVE BUFFER
N	RSTAT	0E8H	RECEIVE STATUS (DMA & GSC)
N	SARL0	0A2H	DMA SOURCE ADDR 0 (LOW)
N	SARH0	0A3H	DMA SOURCE ADDR 0 (HIGH)
N	SARL1	0B2H	DMA SOURCE ADDR 1 (LOW)
N	SARH1	0B3H	DMA SOURCE ADDR 1 (HIGH)
O	SBUF	099H	LOCAL SERIAL CHANNEL (LSC) BUFFER
O	SCON	098H	LOCAL SERIAL CHANNEL (LSC) CONTROL
N	SLOTTM	0B4H	GSC SLOT TIME
O	SP	081H	STACK POINTER
N	TCDCNT	0D4H	GSC TRANSMIT COLLISION COUNTER
O	TCON	088H	TIMER CONTROL
N	TFIFO	085H	GSC TRANSMIT BUFFER
O	TH0	08CH	TIMER 0 (HIGH)
O	TH1	08DH	TIMER 1 (HIGH)
O	TL0	08AH	TIMER 0 (LOW)
O	TL1	08BH	TIMER 1 (LOW)
O	TMOD	089H	TIMER MODE
N	TSTAT	0D8H	TRANSMIT STATUS (DMA & GSC)

TFIFO - (85H) TFIFO is used to access a 3-byte FIFO that contains the transmission data for the GSC.

TSTAT - (0D8H) Contains the DMA Service bit (DMA), Transmit Enable bit (TEN), Transmit FIFO Not Full bit (TFNF), Transmit Done bit (TDN), Transmit Collision Detect bit (TCDT), Underrun bit (UR), No Acknowledge bit (NOACK), and the Receive Data Line Idle bit (LNI). This register is used with both DMA and GSC.

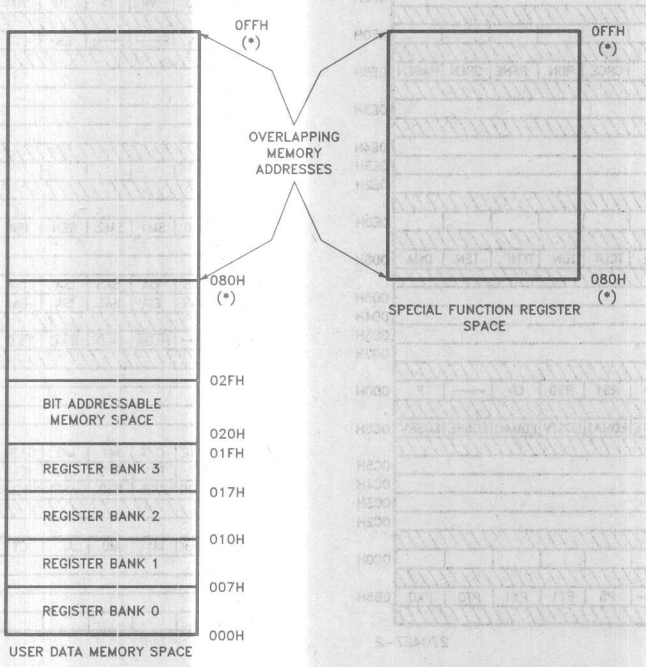
The general purpose flag bits (GF0 and GF1) that exist on the 80C51BH are no longer available on the C152. GF0 has been renamed GFIEN (GSC Flag Idle Enable) and is used to enable idle fill flags. Also GF1 has been renamed XRCLK (External Receive Clock Enable) and is used to enable the receiver to be clocked externally.

2.1.2 DATA MEMORY

Internal data memory consists of 256 bytes as shown in Figure 2.1. The first 128 bytes are addressed exactly like an 80C51BH, using direct addressing.

The addresses of the second 128 bytes of data memory happen to overlap the SFR addresses. The SFRs and their memory locations are shown in Figure 2.2. This means that internal data memory spaces have the same address as the SFR address. However, each type of memory is addressed differently. To access data memory above 80H, indirect addressing or the DMA channels must be used. To access the SFRs, direct addressing is used. When direct addressing is used, the address is the source or destination, e.g. MOV A, 10H, moves the contents of location 10H into the accumulator. When indirect addressing is used, the address of the destination or source exists within another register, e.g. MOV A, @R0. This instruction moves the contents of the memory location addressed by R0 into the accumulator. Directly addressing the locations 80H to 0FFH will access the SFRs. Another form of indirect addressing is with the use of Stack Pointer Operations. If the Stack Pointer contains an address and a PUSH or POP instruction is executed, indirect addressing is actually used. Directly accessing an unused SFR address will give undefined results.

Physically, there are separate SFR memory and data memory spaces allocated on the chip. Since there are separate spaces, the SFRs do not diminish the available data memory space.



*NOTE:
User data memory above 80H must be addressed indirectly. Using direct addressing above 80H accesses the Special Function Registers.

Figure 2.1. Data Memory Map

External data memory is accessed like an 80C51BH, with "MOVX" instructions. Addresses up to 64K may be accessed when using the Data Pointer (DPTR). When accessing external data memory with the DPTR, the address appears on Port 0 and 2. When using the DPTR, if less than 64K of external data memory is used, the address is emitted on all sixteen pins. This means that when using the DPTR, the pins of Port 2 not used for addresses cannot be used for general purpose I/O. An alternative to using 16-bit addresses with the DPTR is to use R0 or R1 to address the external data memory. When using the registers to address external data memory, the address range is limited to 256 bytes. However, software manipulation of I/O Port 2 pins as normal I/O, allows this 256 bytes restriction to be expanded via bank switching. When using R0 or R1 as data pointers, Port 2 pins that are not used for addressing, can be used as general purpose I/O.

2.1.2.1 Bit Addressable Memory

The C152 has several memory spaces in which the bits are directly addressed by their location. The directly addressable bits and their symbolic names are shown in Figure 2.3A, 2.3B, and 2.3C.

Bit addresses 0 to 7FH reside in on-board user data RAM in byte addresses 20H to 2FH (see Figure 2.3A).

Bit addresses 80H to 0FFH reside in the SFR memory space, but not every SFR is bit addressable, see Figure 2.3B. The addressable bits are scattered throughout the SFRs. The addressable bits occur every eighth SFR address starting at 80H and occupy the entire byte. Most of the bits that are addressable in the SFRs have been given symbolic names. These names will often be referred to in this or other documentation on the C152. Most assemblers also allow the use of the symbolic names when writing in assembly language. These names are shown in Figure 2.3C.

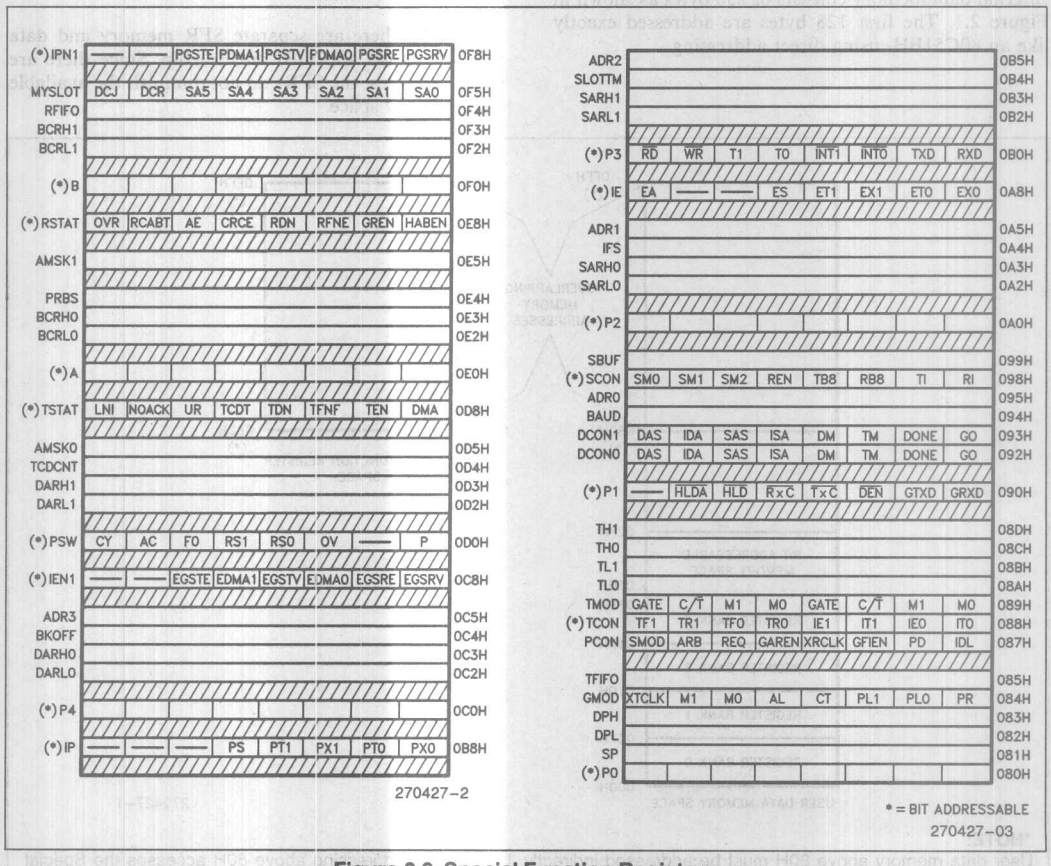


Figure 2.2. Special Function Registers

Data Memory Map (bits):

Byte Address	BIT ADDRESSES (MSB)							(LSB)
020H	07	06	05	04	03	02	01	00
021H	0F	0E	0D	0C	0B	0A	09	08
022H	17	16	15	14	13	12	11	10
023H	1F	1E	1D	1C	1B	1A	19	18
024H	27	26	25	24	23	22	21	20
025H	2F	2E	2D	2C	2B	2A	29	28
026H	37	36	35	34	33	32	31	30
027H	3F	3E	3D	3C	3B	3A	39	38
028H	47	46	45	44	43	42	41	40
029H	4F	4E	4D	4C	4B	4A	49	48
02AH	57	56	55	54	53	52	51	50
02BH	5F	5E	5D	5C	5B	5A	59	58
02CH	67	66	65	64	63	62	61	60
02DH	6F	6E	6D	6C	6B	6A	69	68
02EH	77	76	75	74	73	72	71	70
02FH	7F	7E	7D	7C	7B	7A	79	78

Figure 2.3A. Bit Addresses

Byte Address	BIT ADDRESSES (MSB)							(LSB)	
080H	87	86	85	84	83	82	81	80	(P0)
088H	8F	8E	8D	8C	8B	8A	89	88	(TCON)
090H	97	96	95	94	93	92	91	90	(P1)
098H	9F	9E	9D	9C	9B	9A	99	98	(SCON)
0A0H	A7	A6	A5	A4	A3	A2	A1	A0	(P2)
0A8H	AF	–	–	AC	AB	AA	A9	A8	(IE)
0B0H	B7	B6	B5	B4	B3	B2	B1	B0	(P3)
0B8H	–	–	–	BC	BB	BA	B9	B8	(IP)
0C0H	C7	C6	C5	C4	C3	C2	C1	C0	(P4)
0C8H	–	–	CD	CC	CB	CA	C9	C8	(IEN1)
0D0H	D7	D6	D5	D4	D3	D2	D1	D0	(PSW)
0D8H	DF	DE	DD	DC	DB	DA	D9	D8	(TSTAT)
0E0H	E7	E6	E5	E4	E3	E2	E1	E0	(A)
0E8H	EF	EE	ED	EC	EB	EA	E9	E8	(RSTAT)
0F0H	F7	F6	F5	F4	F3	F2	F1	F0	(B)
0F8H	–	–	FD	FC	FB	FA	F9	F8	(IPN1)

Figure 2.3B. Bit Addresses

Byte Address	SYMBOLIC NAME BIT MAP (MSB)							(LSB)	
080H	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	(P0)
088H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	(TCON)
090H	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	(P1)
098H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	(SCON)
0A0H	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	(P2)
0A8H	EA	—	—	ES	ET1	EX1	ET0	EX0	(IE)
0B0H	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	(P3)
0B8H	—	—	—	PS	PT1	PX1	PT0	PX0	(IP)
0C0H	P4.7	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0	(P4)
0C8H	—	—	EGSTE	EDMA1	EGSTV	EDMA0	EGSRE	EGSRV	(IEN1)
0D0H	CY	AC	F0	RS1	RS0	OV	—	P	(PSW)
0D8H	LNI	NOACK	UR	TCDT	TDN	TFNF	TEN	DMA	(TSTAT)
0E0H									(A)
0E8H	OVR	RCABT	AE	CRCE	RDN	RFNE	GREN	HABEN	(RSTAT)
0F0H									(B)
0F8H	—	—	PGSTE	PDMA1	PGSTV	PDMA0	PGSRE	PGSRV	(IPN1)

Figure 2.3C. Bit Addresses

2.1.3 PROGRAM MEMORY

The 83C152 contains 8K of ROM program memory, and the 80C152 uses only external program memory. Figure 2.4 shows the program memory locations and where they reside. The user is allowed a maximum of 64K of program memory. In the 83C152 program memory fetches beyond 8K automatically access external program memory. When program memory is externally addressed, all of the Port 2 pins emit the address. Since all of Port 2 is affected by the address, unused address pins cannot be used as normal I/O ports even if less than 64K of memory is being accessed.

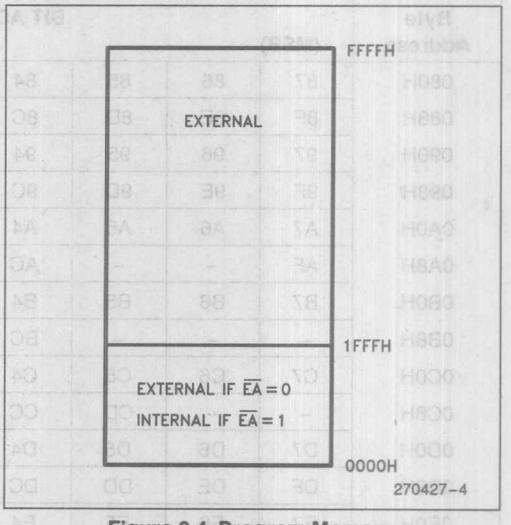


Figure 2.4. Program Memory

2.2 Interrupt Structure

The C152 retains all five interrupts of the 80C51BH. In addition, six new interrupts have been added for a total of 11 available interrupts. Two SFRs have been added to the C152 for control of the new interrupts. These added SFRs are IEN1 (C8H) for enabling the interrupts and IPN1 (F8H) for setting the priority. For an explanation on how the priority of interrupts affects their operation please refer to the MCS-51 Architecture and Hardware Chapters in the Intel Embedded Controller Handbook.

IEN1 FUNCTIONS			
Symbol	**Position**	**Vector**	**Function**
—	IEN1.7		**RESERVED** and do not exist on chip.
—	IEN1.6		**RESERVED** and do not exist on chip.
EGSTE	IEN1.5	04BH	**GSC TRANSMIT ERROR**—If TSTAT.0 (DMA) is cleared, the interrupt service routine at 4BH is invoked when TSTAT.6 (NOACK) or TSTAT.4 (TCDT) is set and EGSTE is enabled. If TSTAT.0 (DMA) is set, the interrupt service routine will be invoked when the TSTAT.5 (UR) is set and EGSTE is enabled.
EDMA1	IEN1.4	053H	**DMA CHANNEL REQUEST 1**—The interrupt service routine at53H is invoked when DCON1.1 (DONE) is set and EDMA1 is enabled.
EGSTV	IEN1.3	043H	**GSC TRANSMIT VALID**—If TSTAT.0 (DMA) is cleared, the interrupt service routine at 43H is invoked when TSTAT.2 (TFNF) is set and EGSTV is enabled. If TSTAT.0 (DMA) is set, the interrupt service routine will be invoked when TSTAT.3 (TDN) is set and EGSTV is enabled.
EDMA0	IEN1.2	03BH	**DMA CHANNEL REQUEST 0**—The interrupt service routine at 3BH will be invoked when DCON0.1 (DONE) is set and EDMA0 is enabled.
EGSRE	IEN1.1	033H	**GSC RECEIVE ERROR**—The interrupt service routine at 33H will be invoked when RSTAT.4 (CRCE), RSTAT.7 (OVR), RSTAT.6 (RCABT), or RSTAT.5 (AE), is set and EGSRE is enabled. This functions the same whether or not TSTAT.0 (DMA) is set.
EGSRV	IEN1.0	02BH	**GSC RECEIVE VALID**—If TSTAT.0 (DMA) is cleared, the interrupt service routine at 2BH will be invoked when RSTAT.2 (RFNE) is set and EGSRV is enabled. If TSTAT.0 (DMA) is set, the interrupt service routine will be invoked when RSTAT.3 (RDN) is set and EGSRV is enabled.

IPN1 is used the same way the current 80C51BH interrupt priority register (IP) is. By assigning a "1" to the appropriate bit, that interrupt has a higher priority than an interrupt with a "0" assigned to it in the priority register.

The new interrupt priority register (IPN1) contents are:

Symbol	Position	Function
PGSTE	IPN1.5	GSC TRANSMIT ERROR
PDMA1	IPN1.4	DMA CHANNEL REQUEST 1
PGSTV	IPN1.3	GSC TRANSMIT VALID
PDMA0	IPN1.2	DMA CHANNEL REQUEST 0
PGSRE	IPN1.1	GSC RECEIVE ERROR
PGSRV	IPN1.0	GSC RECEIVE VALID

registers:

Priority Sequence	Priority Symbolic Address	Priority Symbolic Name	Interrupt Symbolic Address	Interrupt Symbolic Name	Vector Address	
1	IP.0	PX0	IE.0	EX0	03H	(FIRST)
2	IPN1.0	PGSRV	IEN1.0	EGSRV	2BH	
3	IP.1	PT0	IE.1	ET0	0BH	
4	IPN1.1	PGSRE	IEN1.1	EGSRE	33H	
5	IPN1.2	PDMA0	IEN1.2	EDMA0	3BH	
6	IP.2	PX1	IE.2	EX1	13H	
7	IPN1.3	PGSTV	IEN1.3	EGSTV	43H	
8	IPN1.4	PDMA1	IEN1.4	EDMA1	53H	
9	IP.3	PT1	IE.3	ET1	1BH	
10	IPN1.5	PGSTE	IEN1.5	EGSTE	4BH	
11	IP.4	PS	IE.4	ES	23H	(LAST)

2.3 Reset

RESET performs the same operations in both the 80C51BH and the C152 and those conditions that exist at the end of a valid RESET are:

Register	Contents	Register	Contents
ACC	00H	P0–P4	0FFH
ADR0-3	00H	PCON	0XXX0000B
AMSK0	00H	PRBS	00H
AMSK1	00H	PSW	00H
B	00H	RFIFO	INDETERMINATE
BAUD	00H	RSTAT	00000000B
BCRH0	INDETERMINATE	SARH0	INDETERMINATE
BCRH1	INDETERMINATE	SARH1	INDETERMINATE
BCRL0	INDETERMINATE	SARL0	INDETERMINATE
CRL1	INDETERMINATE	SARL1	INDETERMINATE
BKOFF	INDETERMINATE	SBUF	INDETERMINATE
DARH0	INDETERMINATE	SCON	00H
DARH1	INDETERMINATE	SLOTTM	00H
DARL0	INDETERMINATE	SP	07H
DARL1	INDETERMINATE	TCDCNT	INDETERMINATE
DCON0	00H	TCON	00H
DCON1	00H	TFIFO	INDETERMINATE
DPTR	0000H	TH0	00H
GMOD	X0000000B	TH1	00H
IE	0XX00000B	TL0	00H
IEN1	XX000000B	TL1	00H
IFS	00H	TMOD	00H
IP	XXX00000B	TSTAT	XX000100B
IPN1	XX000000B	PC	0000H
MYSLOT	00000000B		

The same conditions apply for both the 80C51BH and C152 for a correct reset pulse or "power-on" reset except that Reset is active low on the C152. Please refer to the 8051/52 Hardware Description Chapter of the Intel Embedded Controller Handbook for an explanation on how to provide a proper power-on reset. Since Reset is active low on the C152, the resistor should be tied to VCC and the capacitor should be tied to VSS.

Because the clocking on part of the GSC circuitry is independent of the processor clock, data may still be transmitted and DEN active for some time after reset is applied. The transmission may continue for a maximum of four machine cycles after reset is first pulled low. Although Reset has to be held low for only three machine cycles to be recognized by the GSC hardware, all of the GSC circuitry may not be reset until four machine cycles have passed. If it is important in the user application that all transmission and DEN becomes inactive at the end of a reset, then Reset will have to be held low for a minimum of four machine cycles.

2.4 Port 4

Port 4 operation is identical to Ports 1-3 on the 80C51BH. The description of port operation can be found in the 8051/52 Hardware Description Chapter of the Intel Embedded Controller Handbook.

2.5 Timer/Counters

The 80C51BH and C152 have the same pair of 16-bit general purpose timer/counters. The user should refer to the Intel Embedded Controller Handbook which describes the timer/counters and their use. The user should bear in mind, when reading the Intel Embedded Controller Handbook that the C152 does not have the third event timer named Timer 2, which is in the 8052.

2.6 Package

The 83C152 is packaged in a 48 pin DIP and a 68 lead PLCC. This differs from the 40 pin DIP and 44 pin PLCC of the 80C51BH. The larger package is required to accommodate the extra 8 bit I/O port (P4). Figure 2.5A and 2.5B show the packages and the pin names.

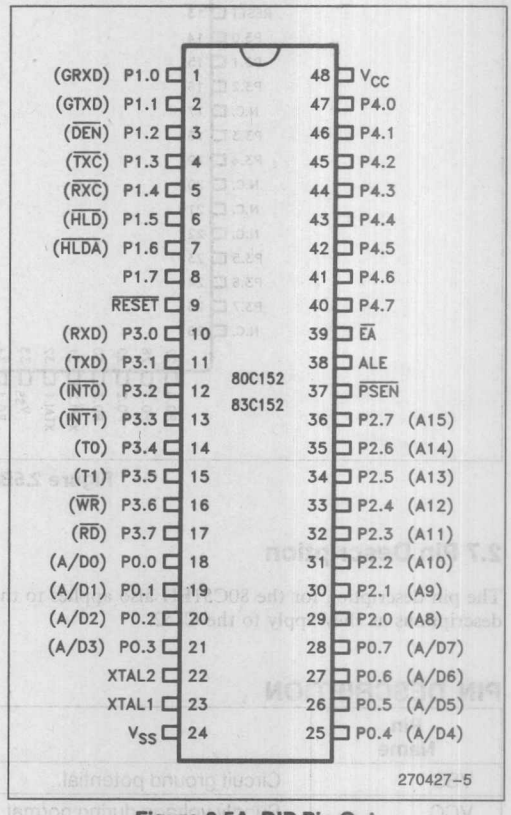


Figure 2.5A. DIP Pin Out

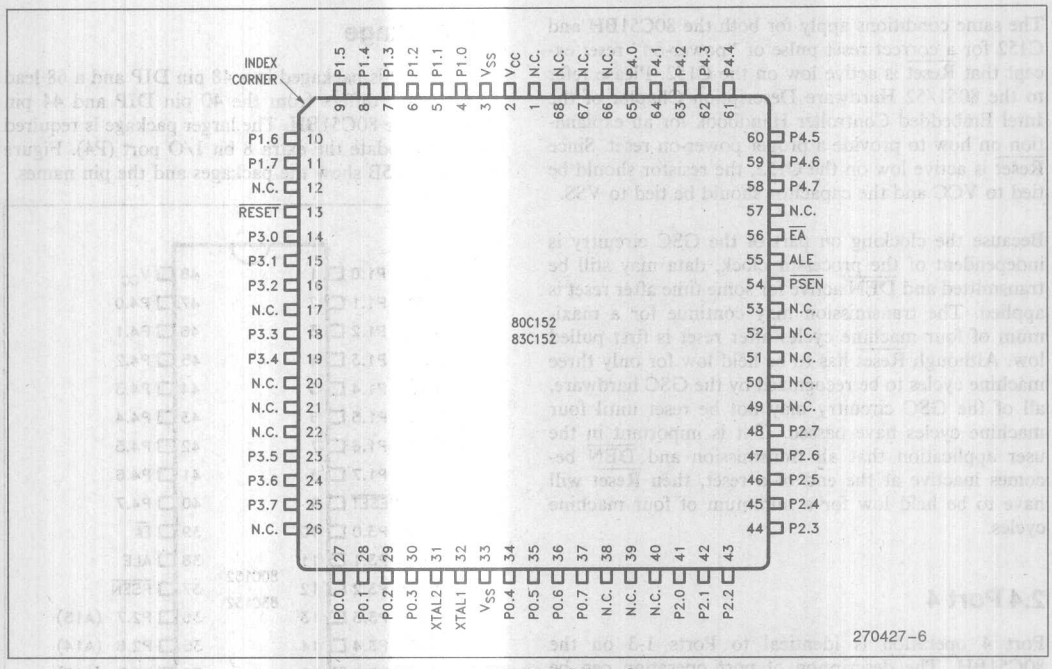


Figure 2.5B. PLCC Pin Out

2.7 Pin Description

The pin description for the 80C51BH also applies to the C152 and is listed below. Changes have been made to the descriptions as they apply to the C152.

PIN DESCRIPTION

Pin Name	Description
VSS	Circuit ground potential.
VCC	Supply voltage during normal, Idle, and Power down operation. Nominally +5V ±10%.
XTAL1	Input to the inverting oscillator amplifier. Also serves as the input for using an external clock signal.
XTAL2	Output from the oscillator amplifier.
PORT 0	Port 0 is an 8-bit open drain bi-directional I/O port. Port 0 pins that have 1s written to them float and in that state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s. Port 0 also outputs the code bytes during program verification in the 83C152. External pullups are required during program verification.
PORT 1	Port 1 is an 8-bit bi-directional I/O port with internal pullups. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current because of the internal pullups. Port 1 also has the following special functions and for the special functions to operate a "1" has to be written to that pin first.

2.7 PIN DESCRIPTION (Continued)

<table>
<tr><th>Pin Name</th><th colspan="2">Description</th></tr>
<tr><td></td><td>Port</td><td>Alternate Function</td></tr>
<tr><td></td><td>1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7</td><td>GSC receiver data input (GR x D)
GSC transmitter data output (GT x D)
Drive Enable to enable external drivers ($\overline{DEN}$)
GSC external transmit clock input ($\overline{T}$ x C)
GSC external receive clock input ($\overline{R}$ x C)
DMA hold request I/O ($\overline{HOLD}$)
DMA hold acknowledge I/O ($\overline{HLDA}$)
none</td></tr>
<tr><td>PORT 2</td><td colspan="2">Port 2 is an 8-bit bi-directional I/O port with internal pullups. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current because of the internal pullups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s and cannot be used as inputs. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 SFR. Port 2 receives the high-order address bits during program verification of the ROM device.</td></tr>
<tr><td>PORT 3</td><td colspan="2">Port 3 is an 8-bit bi-directional I/O port with internal pullups. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally pulled low will source current because of the pullups. Port 3 also has the following special functions and for the special functions to operate that pin must be programmed to a "1" first.</td></tr>
<tr><td></td><td>Port</td><td>Alternate Function</td></tr>
<tr><td></td><td>3.0
3.1
3.2
3.3
3.4
3.5
3.6
3.7</td><td>RXD (LSC serial data input port)
TXD (LSC serial data output port)
$\overline{INT0}$ (external interrupt 0)
$\overline{INT1}$ (external interrupt 1)
T0 (Timer 0 external input)
T1 (Timer 1 external input)
$\overline{WR}$ (external data memory write strobe)
$\overline{RD}$ (external data memory read strobe)</td></tr>
<tr><td>PORT 4</td><td colspan="2">Port 4 is an 8-bit bi-directional I/O port with 40 internal pullups. Port 4 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 4 pins that are externally pulled low will source current because of the pullups. Port 4 also receives the low-order address bytes during program verification in the 83C152.</td></tr>
<tr><td>$\overline{RESET}$</td><td colspan="2">Reset input. A low level on this pin for two machine cycles while the oscillator is running resets the device. An internal diffused resistor to VCC permits Power-On reset using only an external capacitor to VSS.</td></tr>
<tr><td>$\overline{EA}$</td><td colspan="2">External Access enable. $\overline{EA}$ must be externally held low in order to enable the device to fetch code from external Program Memory locations 0000H to 1FFFH.</td></tr>
<tr><td>ALE</td><td colspan="2">Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory. (Including DMAs where no $\overline{RD}$/$\overline{WR}$ generated for internal source/destination.)</td></tr>
<tr><td>$\overline{PSEN}$</td><td colspan="2">Program Store Enable is the read strobe to external Program Memory. When the C152 is executing code from external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external Data Memory.</td></tr>
</table>

2.8 Power Down and Idle

Both of these operations function identically as in the 80C51BH. Application Note 252, "Designing with the 80C51BH" gives an excellent explanation on the use of the reduced power consumption modes. Some of the items not covered in AP-252 are the considerations that are applicable when using the GSC or DMA in conjunction with the power saving modes.

The GSC continues to operate normally in Idle as long as the interrupts are enabled. The interrupts need to be enabled, so that the CPU can service the FIFO's and terminate transmission or reception when appropriate. After servicing the GSC, user software will need to again invoke the Idle command as the CPU does not automatically re-enter the Idle mode after servicing the interrupts.

The GSC does not operate while in Power Down so the steps required prior to entering Power Down become more complicated. The sequence when entering Power Down and the status of the I/O is of major importance in preventing damage to the C152 or other components in the system. Since the only way to exit Power Down is with a Reset, several problem areas become very significant. Some of the problems that merit careful consideration are cases where the Power Down occurs during the middle of a transmission, and the possibility that other stations are not or cannot enter this same mode. The state of the GSC I/O pins becomes critical and the GSC status will need to be saved before power down is entered. There will also need to be some method of identifying to the CPU that the following Reset is probably not a cold start and that other stations on the link may have already been initialized.

The DMA circuitry stops operation in both Idle and Power Down modes. Since operation is stopped in both modes, the process should be similar in each case. Specific steps that need to be taken include: notification to other devices that DMA operation is about to cease for a particular station or network, proper withdrawal from DMA operation, and saving the status of the DMA channels. Again, the status of the I/O pins during Power Down needs careful consideration to avoid damage to the C152 or other components.

Port 4 returns to its input state, which is high level using weak pullup devices.

2.9 Local Serial Channel

The Local Serial Channel (LSC) is the name given to the UART that exists on all MCS-51 devices. The LSC's function and operation is exactly the same as on the 80C51BH. For a description on the use of the LSC, refer to the 8051/52 Hardware Description Chapter in the Intel Embedded Controller Handbook, under Serial Interface.

3.0 GLOBAL SERIAL CHANNEL

3.1 Introduction

The Global Serial Channel (GSC) is a multi-protocol, high performance serial interface targeted for data rates up to 2 MBPS with on-chip clock recovery, and 2.4 MBPS using the external clock options. In applications using the serial channel, the GSC implements the Data Link Layer and Physical Link Layer as described in the ISO reference model for open systems interconnection.

The GSC is designed to meet the requirements of a wide range of serial communications applications and is optimized to implement Carrier-Sense Multi-Access with Collision Detection (CSMA/CD) and Synchronous Data Link Control (SDLC) protocols. The GSC architecture is also designed to provide flexibility in defining non-standard protocols. This provides the ability to retrofit new products into older serial technologies, as well as the development of proprietary interconnect schemes for serial backplane environments.

The versatility of the GSC is demonstrated by the wide range of choices available to the user. The various modes of operation are summarized in Table 3.1. In subsequent sections, each available choice of operation will be explained in detail.

In using Table 3.1, the parameters listed vertically (on the left hand side) represent an option that is selected (X). The parameters listed horizontally (along the top of the table) are all the parameters that could theoretically be selected (Y). The symbol at the junction of both X and Y determines the applicability of the option Y.

Note, that not all combinations are backwards compatible. For example, Manchester encoding requires half duplex, but half duplex does not require Manchester encoding.

N = NOT AVAILABLE M = MANDATORY O = OPTIONAL P = NORMALLY PREFERRED X = N/A	DATA ENCODING			FLAGS		CRC			DU-PLEX		ACKNOW-LEDGE			ADDRESS RECOG-NITION			BACKOFF			PRE-AMBLE	
	MANCHESTER	NRZ	NRZI	01111110	11/IDLE	NONE	16BITCCIT	32BITAUTO	HALF	FULL	NONE	HARDWARE	USERDEFINED	NONE/ALL	8BIT	16BIT	NORMAL	ALTERNATE	DETERMINISTIC	NONE	8BIT
DATA ENCODING:																					
MANCHESTER(CSMA/CD)	X	N	N	1	P	1	O	O	M	N	O	O	O	O	O	O	O	O	O	N	O
NRZI (SDLC)	N	X	N	P	1	1	O	O	O	O	O	N	P	O	O	O	N	N	N	O	O
NRZ (EXT CLK)	N	N	X	O	O	1	O	O	O	O	O	N	O	O	O	O	O	O	O	O	O
FLAGS:01111110 (SDLC)	N	P	O	X	1	1	O	O	O	O	O	N	P	O	O	O	N	N	N	O	O
11/IDLE	P	N	O	1	X	1	O	O	O	N	O	O	O	O	O	O	O	O	O	1	O
CRC:NONE	1	1	1	1	1	X	N	N	1	N	1	1	1	1	1	1	N	N	N	1	1
16-BIT CCITT	O	O	O	O	O	N	X	N	O	O	O	O	O	O	O	O	O	O	O	O	O
32-BIT AUTODIN II	O	O	O	O	O	N	N	X	O	N	O	O	O	O	O	O	O	O	O	O	O
DUPLEX:HALF	O	O	O	O	O	1	O	O	X	N	O	O	O	O	O	O	O	O	O	O	O
FULL	N	O	O	M	N	N	M	N	N	X	O	N	P	O	O	O	N	N	N	O	O
ACKNOWLEDGEMENT:NONE	O	O	O	O	O	1	O	O	O	O	X	N	N	O	O	O	O	O	O	O	O
HARDWARE	O	N	N	N	O	1	O	O	O	N	N	X	N	O	O	O	N	O	O	N	O
USER DEFINED	O	P	O	O	O	1	O	O	O	P	N	N	X	O	O	O	O	O	O	O	O
ADDRESS RECOGNITION:																					
NONE/ALL	O	O	O	O	O	1	O	O	O	O	O	O	O	X	N	N	O	O	O	O	O
8-BIT	O	O	O	O	O	1	O	O	O	O	O	O	O	N	X	N	O	O	O	O	O
16-BIT	O	O	O	O	O	1	O	O	O	O	O	O	O	N	N	X	O	O	O	O	O
COLLISION RESOLUTION:																					
NORMAL	O	N	O	N	O	N	O	O	M	N	O	N	O	O	O	O	X	N	N	N	O
ALTERNATE	O	N	O	N	O	N	O	O	M	N	O	O	O	O	O	O	N	X	N	N	O
DETERMINISTIC	O	N	O	N	O	N	O	O	M	N	O	O	O	O	O	O	N	N	X	N	O
PREAMBLE:NONE	N	O	O	O	1	1	O	O	O	O	O	N	O	O	O	O	N	N	N	X	N
8-BIT	O	O	O	O	O	1	O	O	O	O	O	O	O	O	O	O	O	O	O	N	X
32-BIT	O	O	O	O	O	1	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N
64-BIT	O	O	O	O	O	1	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N
JAM:D.C.	M	N	N	N	O	N	O	O	M	N	O	O	O	O	O	O	O	O	O	N	O
$\overline{CRC}$	M	N	N	N	O	N	O	O	M	N	O	O	O	O	O	O	O	O	O	N	O
CLOCKING:EXTERNAL	N	M	N	O	O	N	O	O	O	O	O	N	O	O	O	O	N	N	N	O	O
INTERNAL	O	O	N	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
CONTROL: CPU	O	O	O	O	O	1	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
DMA	O	O	O	O	O	1	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
RAW RECEIVE:	1	1	1	1	1	1	1	1	1	N	1	1	1	1	1	1	O	O	O	1	1
RAW TRANSMIT:	1	1	1	1	1	1	1	1	1	N	1	1	1	1	1	1	N	N	N	1	1
CSMA/CD:	O	N	2	1	P	1	O	O	M	N	O	O	O	O	O	O	O	O	O	N	O
SDLC:	N	O	O	P	1	1	O	O	O	O	O	N	O	O	O	O	N	N	N	P	O

Table 3.1 (Continued)

N = NOT AVAILABLE M = MANDATORY O = OPTIONAL P = NORMALLY PREFERRED X = N/A	PRE-AMBLE		JAM		CLOCK		CONTROL					
	32 BIT	64 BIT	DC	CRC/	EXTERNAL	INTERNAL	CPU	DMA	RAW RECEIVE	RAW TRANSMIT	CSMA/CD	SDLC
DATA ENCODING:												
MANCHESTER	O	O	O	O	N	M	O	O	O	O	M	N
NRZI	O	O	N	N	N	M	O	O	O	O	N	M
NRZ	O	O	O	O	M	N	O	O	O	O	O	O
FLAGS:01111110	O	O	N	N	O	O	O	O	O	1	1	P
11/IDLE	O	O	O	O	O	O	O	O	O	1	P	1
CRC:NONE	1	1	N	N	1	1	1	1	1	1	1	1
16-BIT CCITT	O	O	O	O	O	O	O	O	1	1	O	O
32-BIT AUTODIN II	O	O	O	O	O	O	O	O	1	1	O	O
DUPLEX:HALF	O	O	O	O	O	O	O	O	O	O	O	O
FULL	O	O	N	N	O	O	O	O	N	N	N	P
ACKNOWLEDGEMENT:NONE	O	O	O	O	O	O	O	O	O	O	O	O
HARDWARE	O	O	O	O	N	O	O	O	N	N	O	N
USER DEFINED	O	O	O	O	O	O	O	O	O	O	O	1
ADDRESS RECOGNITION:												
NONE	O	O	O	O	O	O	O	O	O	O	O	O
8-BIT	O	O	O	O	O	O	O	O	1	1	O	O
16-BIT	O	O	O	O	O	O	O	O	1	1	O	O
COLLISION RESOLUTION:												
NORMAL	O	O	O	O	N	O	O	O	O	N	M	N
ALTERNATE	O	O	O	O	N	O	O	O	O	N	M	N
DETERMINISTIC	O	O	O	O	N	O	O	O	O	N	M	N
PREAMBLE:NONE	N	N	N	N	O	O	O	O	O	O	N	P
8-BIT	N	N	O	O	O	O	O	O	1	1	O	O
32-BIT	X	N	O	O	O	O	O	O	1	1	O	O
64-BIT	N	X	O	O	O	O	O	O	1	1	O	O
JAM:D.C.	O	O	X	N	2	O	O	O	O	N	M	N
$\overline{\text{CRC}}$	O	O	N	X	2	O	O	O	O	N	M	N
CLOCKING:EXTERNAL	O	O	N	N	X	N	O	O	O	O	2	O
INTERNAL	O	O	O	O	N	X	O	O	O	O	O	O
CONTROL:CPU	O	O	O	O	O	O	X	N	O	O	O	O
DMA	O	O	O	O	O	O	N	X	O	O	O	O
RAW RECEIVE:	1	1	O	O	1	1	1	1	X	N	1	1
RAW TRANSMIT:	1	1	N	N	1	1	1	1	N	X	1	1
CSMA/CD:	O	O	O	O	2	O	O	O	O	O	X	N
SDLC:	O	O	N	N	O	O	O	O	O	O	N	X

Note 1: Programmable in Raw transmit or receive mode.

Note 2: When CSMA/CD is enabled, an external clock can be used on the transmitter, but not the receiver. Since the receiver monitors the link for Manchester violations, external hardware would be required to reformat the data from NRZ to Manchester on the transmitter. These hardware requirements go beyond the expectations of this table for implementation. For that reason it was assumed that the external clock cannot be used at all with CSMA/CD protocol, although it is actually possible to do so.

Almost all the options available from Table 3.1 can be implemented with the proper software to perform the functions that are necessary for the options selected. In Table 3.1, a judgment has been made by the authors on which options are practical and which are not. What this means is that in Table 3.1, an "N" should be interpreted as meaning that the option is either not practical when implemented with user software or that it cannot be done. An "O" is used when that function is one of several that can be implemented with the GSC without additional user software.

The GSC is targeted to operate at bit rates up to 2.4 MBps using the external clock options and up to 2 MBps using the internal baud rate generator, internal data formatting and on-chip clock recovery. The baud rate generator allows most standard rates to be achieved. These standards include the proposed IEEE802.3 LAN standard (1.0MBps) and the T1 standard (1.544MBps). The baud rate is derived from the crystal frequency. This makes crystal selection important when determining the frequency and accuracy of the baud rate.

3.2 CSMA/CD Operation

3.2.1 CSMA/CD OVERVIEW

CSMA/CD operates by sensing the transmission line for a carrier, which indicates link activity. At the end of link activity, a station must wait a period of time, called the deference period, before transmission may begin. The deference period is also known as the interframe space. The interframe space is explained in Section 3.2.3.

With this type of operation, there is always the possibility of a collision occurring after the deference period due to line delays. If a collision is detected after transmission is started, a jamming mechanism is used to ensure that all stations monitoring the line are aware of the collision. A resolution algorithm is then executed to resolve the contention. There are three different modes of collision resolution made available to the user on the C152. Re-transmission is attempted when a resolution algorithm indicates that a station's opportunity has arrived.

Normally, in CSMA/CD, re-transmission slot assignments are intended to be random. This method gives all stations an equal opportunity to utilize the serial communication link but also leaves the possibility of another collision due to two stations having the same slot assignment. There is an option on the C152 which allows all the stations to have their slot assignments previously determined by user software. This pre-assignment of slots is called the deterministic resolution mode. This method allows resolution after the first collision and ensures the access of the link to each station during the resolution. Deterministic resolution can be advantageous when the link is being heavily used and collisions are frequently occurring and in real time applications where determinism is required. Deterministic resolution may also be desirable if it is known beforehand that a certain station's communication needs to be prioritized over those of other stations if it is involved in a collision.

3.2.2 CSMA/CD FRAME FORMAT

The frame format in CSMA/CD consists of a preamble, Beginning of Frame flag (BOF), address field, information field, CRC, and End of Frame flag (EOF) as shown in Figure 3.1.



Figure 3.1 Typical CSMA/CD Frame

PREAMBLE - The preamble is a series of alternating 1s and 0s. The length of the preamble is programmable to be 0, 8, 32, or 64 bits. The purpose of the preamble is to allow all the receivers to synchronize to the same clock edges and identifies to the other stations on-line that there is activity indicating the link is being used. For these reasons zero preamble length is not compatible with standard CSMA/CD, protocols. When using CSMA/CD, the BOF is considered part of the preamble compared to SDLC, where the BOF is not part of the preamble. This means that if zero preamble length were to be used in CSMA/CD mode, no BOF would be generated. It is strongly recommended that zero preamble length never be used in CSMA/CD mode. If the preamble contains two consecutive 0s, the preamble is considered invalid. If the C152 detects an invalid preamble, the frame is ignored.

BOF - In CSMA/CD the Beginning-Of-Frame is a part of the preamble and consists of two sequential 1s. The purpose of the BOF is to identify the end of the preamble and indicate to the receiver(s) that the address will immediately follow.

ADDRESS - The address field is used to identify which messages are intended for which stations. The user must assign addresses to each destination and source. How the addresses are assigned, how they are maintained, and how each transmitter is made aware of which addresses are available is an issue that is left to the user. Some suggestions are discussed in Section 3.5.5. Generally, each address is unique to each station but there are special cases where this is not true. In these special cases, a message is intended for more than one station. These multi-targeted messages are called broadcast or multicast-group addresses. A broadcast address consisting of all 1s will always be received by all stations. A multicast-group address usually is indicated by using a 1as the first address bit. The user can choose to mask off all or selective bits of the address so that the GSC receives all messages or multicast-group messages. The address length is programmable to be 8 or 16 bits. An address consisting of all 1s will always be received by the GSC on the C152. The address bits are always passed from the GSC to the CPU. With user software, the address can be extended beyond 16 bits, but the automatic address recognition will only work on a maximum of 16 bits. User software will have to resolve any remaining address bits.

INFO - This is the information field and contains the data that one device on the link wishes to transmit to another device. It can be of any length the user wishes but needs to be in multiples of 8 bits. This is because multiples of 8 bits are used to transfer data into or out of the GSC FIFOs. The information field is delineated from the rest of the components of the frame by the preceding address field and the following CRC. The receiver determines the position of the end of the information field by passing the bytes through a temporary storage space. When the EOF is received the bytes in temporary storage are the CRC, and the last bit received previous to the CRC constitute the end of the information field.

CRC - The Cyclic Redundancy Check (CRC) is an error checking algorithm commonly used in serial communications. The C152 offers two types of CRC algorithms, a 16-bit and a 32-bit. The 16-bit algorithm is normally used in the SDLC mode and will be described in the SDLC section. In CSMA/CD applications either algorithm can be used but IEEE 802.3 uses a 32-bit CRC. The generation polynomial the C152 uses with the 32-bit CRC is:

$$G(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

The CRC generator, as shown in Figure 3.2, operates by taking each bit as it is received and XOR'ing it with bit 31 of the current CRC. This result is then placed in temporary storage. The result of XOR'ing bit 31 with the received bit is then XOR'd with bits 0, 1, 3, 4, 6, 7, 9, 10, 11, 15, 21, 22, 25 as the CRC is shifted right one position. When the CRC is shifted right, the temporary storage space holding the result of XOR'ing bit 31 and the incoming bit is shifted into position 0. The whole process is then repeated with the next incoming or outgoing bit.

The user has no access to the CRC generator or the bits which constitute the CRC while in CSMA/CD. On transmission, the CRC is automatically appended to the data being sent, and on reception, the CRC bits are not normally loaded into the receive FIFO. Instead, they are automatically stripped. The only indication the user has for the status of the CRC is a pass/fail flag. The pass/fail flag only operates during reception. A CRC is considered as passing when the the CRC generator has 11000111 00000100 11011010 01111011B as a remainder after all of the data, including the CRC checksum, from the transmitting station has been cycled through the CRC generator. The preamble, BOF and EOF are not included as part of the CRC algorithm. An interrupt is available that will interrupt the CPU if the CRC of the receiver is invalid. The user can enable the CRC to be passed to the CPU by placing the receiver in the raw receive mode.

This method of calculating the CRC is compatible with IEEE 802.3.

EOF - The End Of Frame indicates when the transmission is completed. The end flag in CSMA/CD consists of an idle condition. An idle condition is assumed when there is no transitions and the link remains high for 2 or more bit times.

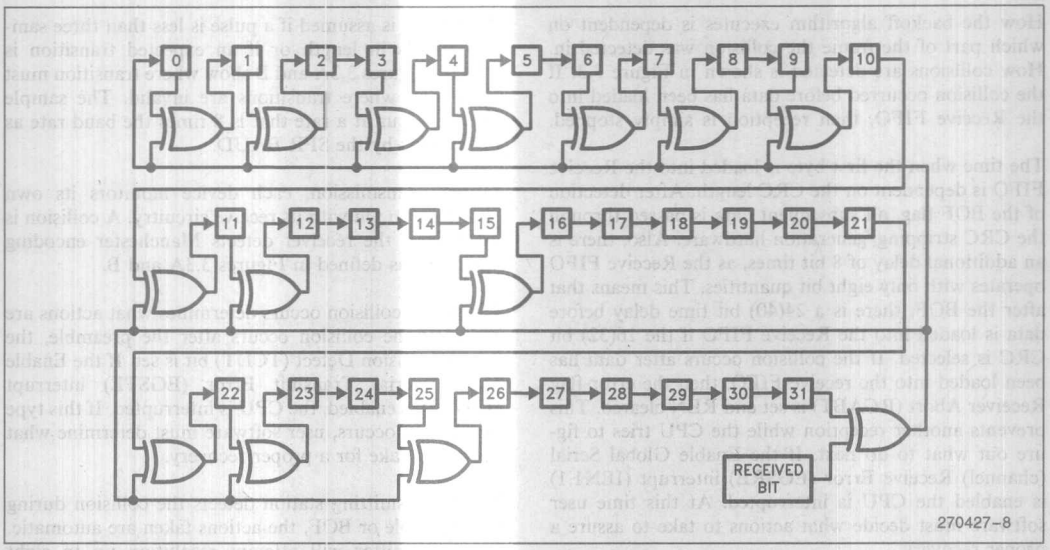


Figure 3.2. CRC Generator

3.2.3 INTERFRAME SPACE

The interframe space is the amount of time that transmission is delayed after the link is sensed as being idle and is used to separate transmitted frames. In alternate backoff mode, the interframe space may also be included in the determination of when retransmissions may actually begin. The C152 allows programmable interframe spaces of even numbers of bit times from 2 to 256.

The period of the interframe space is determined by the contents of IFS. IFS is an SFR that is programmable from 0 to 254. The interframe space is measured in bit times. The value in IFS multiplied by the bit time equals the interframe space unless IFS equals 0. If IFS does equal 0, then the interframe space will equal 256 bit times. One of the considerations when loading the IFS is that only even numbers (LSB must be 0) can be used because only the 7 most significant bits are loaded into IFS. The LSB is controlled by the GSC and determines which half of the IFS is currently being used. In some modes, the interframe space timer is re-triggered if activity is detected during the first half of the period. The GSC determines which half of the interframe space is currently being used by examining the LSB. A one indicates the first half and zero indicates the second half of the IFS.

After reset IFS is 0, which delays the first transmission for both SDLC and CSMA/CD by 256 bit times (after reset, a bit time equals 8 oscillator clock periods).

In most applications, the period of the interframe space will be equal to or greater than the amount of time needed to turn-around the received frame. The turn-around period is the amount of time that is needed by user software to complete the handling of a received frame and be prepared to receive the next frame. An interframe space smaller than the required turn-around period could be used, but would allow some frames to be missed.

When a GSC transmitter has a new message to send, it will first sense the link. If activity is detected, transmission will be deferred to allow the frame in progress to complete. When link activity ceases, the station continues deferring for one interframe space period.

As mentioned earlier, the interframe space is used during the collision resolution period as well as during normal transmission. The backoff method selected affects how the deference period is handled during normal transmission. If normal backoff mode is selected, the interframe space timer is reset if activity occurs during approximately the first half of the interframe space. If alternate backoff or deterministic backoff is selected, the timer is not reset. In all cases when the interframe space timer expires, transmission may begin, regardless if there is activity on the link or not. Although the C152 resets the interframe space timer if activity is detected during the first one-half of the interframe space, this is not necessarily true of all CSMA/CD systems. (IEEE 802.3 recommends that the interframe space be reset if activity is detected during the first two-thirds or less of the interframe space.)

3.2.4 COLLISION RESOLUTION

The method used to resolve a collision is called the backoff algorithm.

How the backoff algorithm executes is dependent on which part of the frame the collision was detected in. How collisions are detected is shown in Figure 3.3. If the collision occurred before data has been loaded into the Receive FIFO, then reception is simply stopped.

The time when the first byte is loaded into the Receive FIFO is dependent on the CRC length. After detection of the BOF flag, all subsequent data is passed through the CRC stripping/generation hardware. Also, there is an additional delay of 8 bit times, as the Receive FIFO operates with only eight bit quantities. This means that after the BOF, there is a 24(40) bit time delay before data is loaded into the Receive FIFO if the 16(32) bit CRC is selected. If the collision occurs after data has been loaded into the receive FIFO, then the error flag Receiver Abort (RCABT) is set and REN cleared. This prevents another reception while the CPU tries to figure out what to do next. If the Enable Global Serial (channel) Receive Error (EGSRE) interrupt (IEN1.1) is enabled the CPU is interrupted. At this time user software must decide what actions to take to assure a proper recovery.

A collision is assumed if a pulse is less than three sample periods in length or if an expected transition is missed. Figures 3.3A and B show where transition must occur and where transitions are invalid. The sample periods occur at a rate that is 8 times the baud rate as determined by the SFR BAUD.

During transmission, each device monitors its own transmission pin with its receive circuitry. A collision is assumed if the receiver detects Manchester encoding violations as defined in Figures 3.3A and B.

Where the collision occurs determines what actions are taken. If the collision occurs after the preamble, the Timer Collision Detect (TCDT) bit is set. If the Enable Global Serial Transmit Error (EGSTE) interrupt (IEN1.5) is enabled, the CPU is interrupted. If this type of collision occurs, user software must determine what actions to take for a proper recovery.

If the transmitting station detects the collision during the preamble or BOF, the actions taken are automatic. The transmitter will attempt resolution up to eight tries. After the eighth attempt, the error flag (TCDT) is set. If EGSTE is enabled, the CPU is then interrupted.

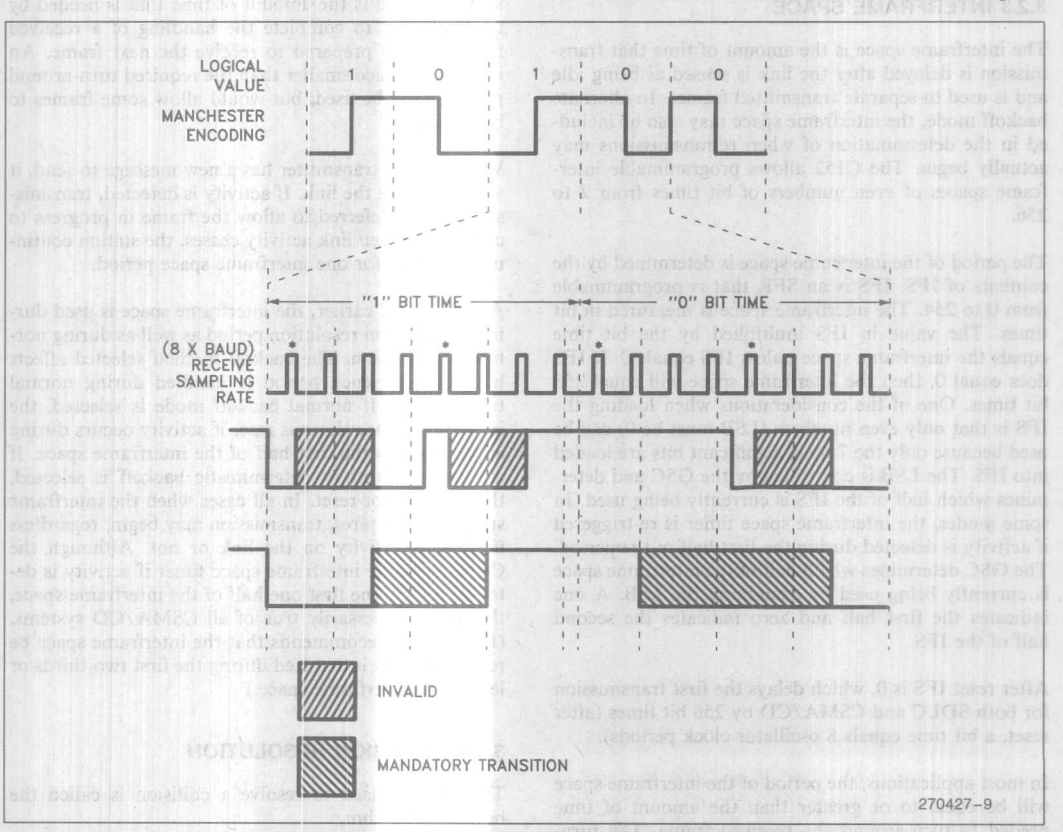


Figure 3.3A. CSMA/CD Collisions

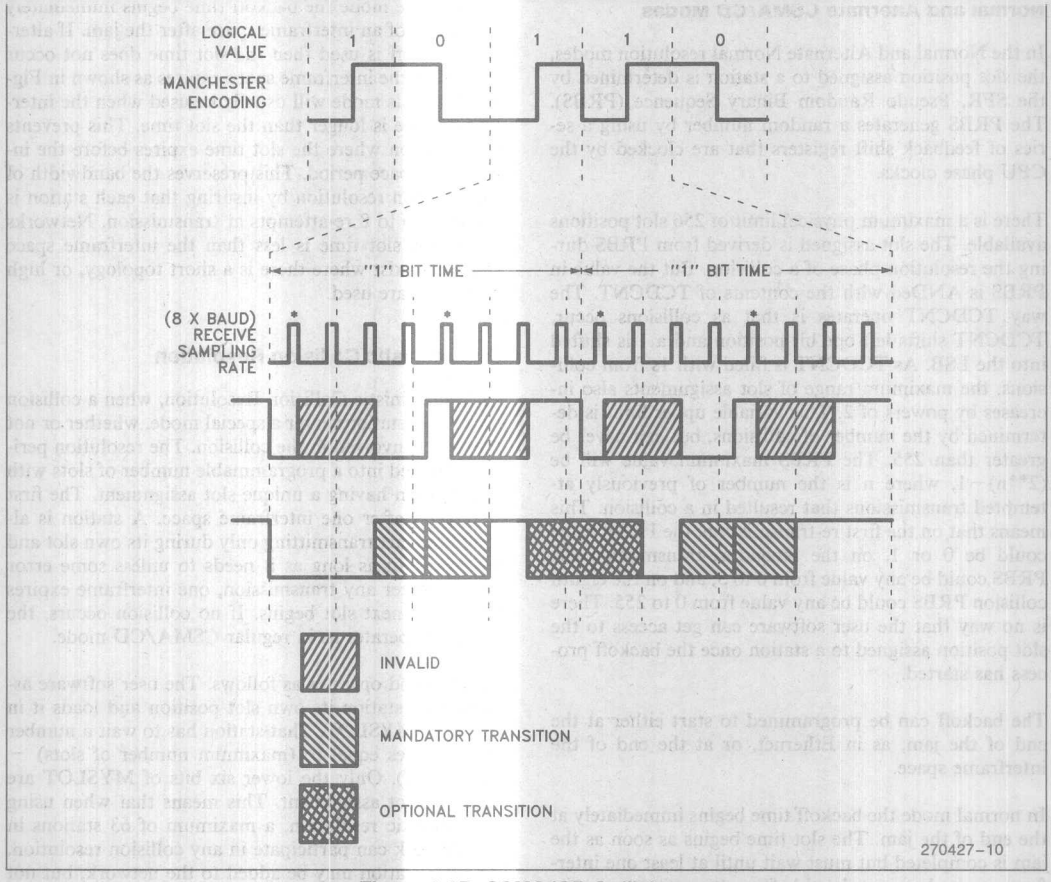


Figure 3.3B. CSMA/CD Collisions

When a transmitting GSC detects a collision, the first action taken is to apply a jamming signal to the link. The jam is sent following the end of the preamble, or immediately if the preamble has already been completed. This action is taken to insure that other stations on the link are aware that a collision has occurred.

The jamming signal can be one of two types, D.C. or $\overline{\text{CRC}}$. D.C. jam is selected by setting the DCJ bit (MYSLOT.7). The D.C. jam applies a continuous low level signal for a duration equal to the $\overline{\text{CRC}}$ length. To select the $\overline{\text{CRC}}$ type of jam, the DCJ bit needs to be cleared. $\overline{\text{CRC}}$ is selected after the C152 is reset. The $\overline{\text{CRC}}$ jam operates by taking the current CRC calculated up to that point, inverting the data and applying that signal to the link.

After applying the jam, the resolution phase is next. This phase will effect the throughput and efficiency of the link once a collision is detected. There are three methods to choose from that determine which backoff algorithm is used. These methods are named: "Normal Backoff", "Alternate Backoff", and "Deterministic Backoff".

Before going into detail on the various backoff schemes, there is a parameter called the slot time that must be understood by the user. The slot time is used during the collision resolution and is the basic scheduling quantum for retransmission once a collision is detected. The slot time also represents the maximum length of a collision fragment and the upper bound on the acquisition time of the network. The value of the slot time is determined by the contents of SLOTTM. SLOTTM is programmable from 0 to 255. A slot time is equal to 256-SLOTTM multiplied by the bit time, unless SLOTTM equals 0. If 0 is used in SLOTTM, then the slot time period will equal 256 bit times. A bit time is equal to 1/baud rate. The timing requirements on the slot time is that it be equal to, or greater than the longest round-trip propagation time of the signal plus the jam time. The jam time is equal to the CRC length.

Normal and Alternate CSMA/CD Modes

In the Normal and Alternate Normal resolution modes, the slot position assigned to a station is determined by the SFR, Pseudo Random Binary Sequence (PRBS). The PRBS generates a random number by using a series of feedback shift registers that are clocked by the CPU phase clocks.

There is a maximum physical limit of 256 slot positions available. The slot assigned is derived from PRBS during the resolution phase of a collision. But the value in PRBS is ANDed with the contents of TCDCNT. The way TCDCNT operates is that as collisions occur, TCDCNT shifts left one bit position and a 1 is shifted into the LSB. As TCDCNT is filled with 1s from collisions, the maximum range of slot assignments also increases by powers of 2. This variable upper limit is determined by the number of collisions, but can never be greater than 255. The PRBS maximum value will be (2**n) − 1, where n is the number of previously attempted transmissions that resulted in a collision. This means that on the first re-transmission, the PRBS value could be 0 or 1, on the second re-transmission the PRBS could be any value from 0 to 3, and on the eighth collision PRBS could be any value from 0 to 255. There is no way that the user software can get access to the slot position assigned to a station once the backoff process has started.

The backoff can be programmed to start either at the end of the jam, as in Ethernet, or at the end of the interframe space.

In normal mode the backoff time begins immediately at the end of the jam. The slot time begins as soon as the jam is completed but must wait until at least one interframe space has completed before attempting transmission. Slot 0 is the first to occur, followed by Slot 1 and so on. This means the lowest slot number assigned will win control of the link as long as the slot time ends after the the interframe space expires. In networks where the slot time is longer than the interframe space, normal backoff will usually be implemented. This is because the interframe space time will expire before Slot 0 is complete. This is shown in Figure 3.4. In networks where the interframe space is longer than the slot time, Slot 0 will expire before the interframe space and will not be able to transmit during that resolution attempt. Taken to an extreme where the interframe space is much larger than the slot time, it is possible that there will be no resolution during the first couple of collisions because the possible number of slot times available will still be less than the interframe space. This would waste the bandwidth of the resolution by not allowing all stations the opportunity of 8 attempts at retransmission.

In alternate mode the backoff time begins immediately at the end of an interframe space after the jam. If alternate backoff is used then the slot time does not occur until after the interframe space expires as shown in Figure 3.4. This mode will usually be used when the interframe space is longer than the slot time. This prevents the situation where the slot time expires before the interframe space period. This preserves the bandwidth of the collision resolution by insuring that each station is allowed up to 8 re-attempts at transmission. Networks where the slot time is less than the interframe space generally exist where there is a short topology, or high data rates are used.

Deterministic Collision Resolution

In Deterministic Collision Resolution, when a collision occurs, all stations enter a special mode, whether or not they were involved in the collision. The resolution period is divided into a programmable number of slots with each station having a unique slot assignment. The first slot starts after one interframe space. A station is allowed to start transmitting only during its own slot and will transmit as long as it needs to unless some error occurs. After any transmission, one interframe expires before the next slot begins. If no collision occurs, the protocol operates as in regular CSMA/CD mode.

This method operates as follows. The user software assigns each station its own slot position and loads it in the SFR, MYSLOT. That station has to wait a number of slot times equal to (maximum number of slots) − (MYSLOT). Only the lower six bits of MYSLOT are used for slot assignment. This means that when using deterministic resolution, a maximum of 63 stations in the network can participate in any collision resolution. Another station may be added to the network, but not allowed to participate in a resolution. That station should have 0 as its assigned slot. This prevents the station from attempting to retransmit during the collision resolution. When using deterministic resolution, the PRBS must be disabled. By writing 0FFH into the PRBS, it is frozen into an all 1s state. The maximum number of stations that may be involved in the resolution is loaded into TCDCNT. The slot count does not begin until the receiver senses that the line is idle and one interframe space expires. At the end of the interframe space the GSC starts counting the slots. Alternate backoff mode should be selected whenever deterministic backoff mode is being used. Then, if activity is detected on the link with deterministic backoff mode selected, the GSC first waits until the link is idle and then waits until the end of the interframe space before the slot timer continues counting the slots. This allows another station to transmit a pending message in its proper slot and each station gets an opportunity to transmit when the slot time equals the value in its MYSLOT register.

The bottom diagram of Figure 3.4 illustrates this mechanism for a number of stations (MAX). It shows stations with slot numbers (MAX-2), 2 and 1 transmitting during the resolution period. Note the interframe space after the jam and after each transmission. After all the slot positions have been exercised, normal CSMA/CD operation resumes. Unless deterministic resolution is selected, it is possible that stations not involved in the collision may attempt transmission during the resolution period after an interframe space period. Deterministic mode prevents this from happening because all stations on the link are required to enter into the resolution phase, whether or not they are involved in the collision. Then, a station is either allowed to transmit during its assigned slot or is prevented from transmitting during the resolution period.

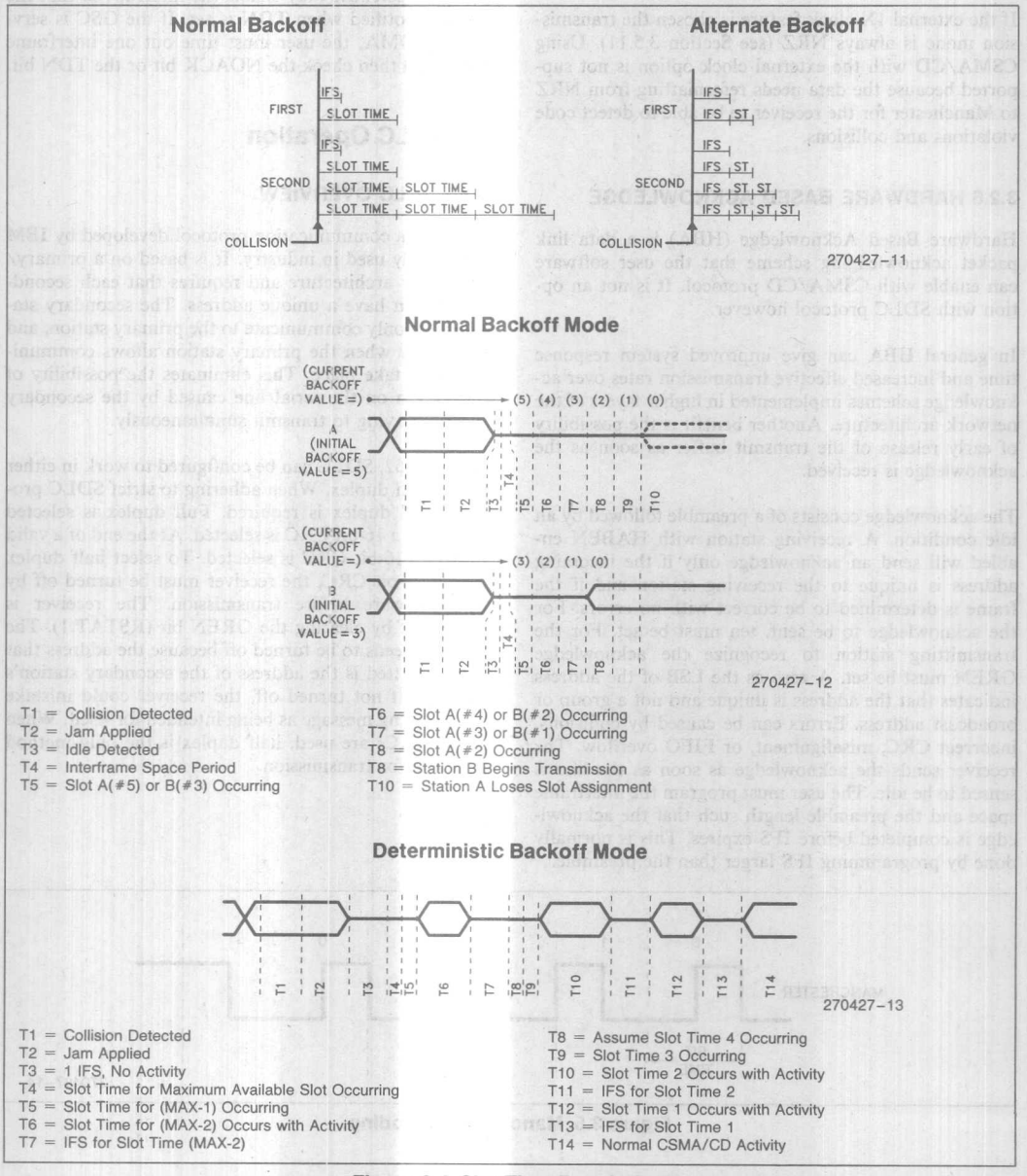


Figure 3.4. Slot Time Resolution

3.2.5 CSMA/CD DATA ENCODING

Manchester encoding/decoding is automatically selected when the user software selects CSMA/CD transmission mode (See Figure 3.5). In Manchester encoding the value of the bit is determined by the transition in the middle of the bit time, a positive transition is decoded as a 1 and a negative transition is decoded as a 0.

If the external 1X clock feature is chosen the transmission mode is always NRZ (see Section 3.5.11). Using CSMA/CD with the external clock option is not supported because the data needs reformatting from NRZ to Manchester for the receiver to be able to detect code violations and collisions.

3.2.6 HARDWARE BASED ACKNOWLEDGE

Hardware Based Acknowledge (HBA) is a data link packet acknowledging scheme that the user software can enable with CSMA/CD protocol. It is not an option with SDLC protocol however.

In general HBA can give improved system response time and increased effective transmission rates over acknowledge schemes implemented in higher layers of the network architecture. Another benefit is the possibility of early release of the transmit buffer as soon as the acknowledge is received.

The acknowledge consists of a preamble followed by an idle condition. A receiving station with HABEN enabled will send an acknowledge only if the incoming address is unique to the receiving station and if the frame is determined to be correct with no errors. For the acknowledge to be sent, ten must be set. For the transmitting station to recognize the acknowledge GREN must be set. A zero as the LSB of the address indicates that the address is unique and not a group or broadcast address. Errors can be caused by collisions, incorrect CRC, misalignment, or FIFO overflow. The receiver sends the acknowledge as soon as the line is sensed to be idle. The user must program the interframe space and the preamble length such that the acknowledge is completed before IFS expires. This is normally done by programming IFS larger than the preamble.

A transmitting station with HABEN enabled expects an acknowledge. It must receive one prior to the end of the interframe space, or else an error is assumed and the NOACK bit is set. Setting of the TDN bit is also delayed until the end of the interframe space. Collisions detected during the interframe space will also cause NOACK to be set.

The user software may enable the interrupt so that the CPU is notified when TDN is set. If the GSC is serviced by DMA, the user must time out one interframe space and then check the NOACK bit or the TDN bit.

3.3 SDLC Operation

3.3.1 SDLC OVERVIEW

SDLC is a communication protocol developed by IBM and widely used in industry. It is based on a primary/secondary architecture and requires that each secondary station have a unique address. The secondary stations can only communicate to the primary station, and then, only when the primary station allows communication to take place. This eliminates the possibility of contention on the serial line caused by the secondary station's trying to transmit simultaneously.

In the C152, SDLC can be configured to work in either full or half duplex. When adhering to strict SDLC protocol, full duplex is required. Full duplex is selected whenever a 16-bit CRC is selected. At the end of a valid reset the 16-bit CRC is selected. To select half duplex with a 16-bit CRC, the receiver must be turned off by user software before transmission. The receiver is turned off by clearing the GREN bit (RSTAT.1). The receiver needs to be turned off because the address that is transmitted is the address of the secondary station's receiver. If not turned off, the receiver could mistake the outgoing message as being intended for itself. When 32-bit CRCs are used, half duplex is the only method available for transmission.

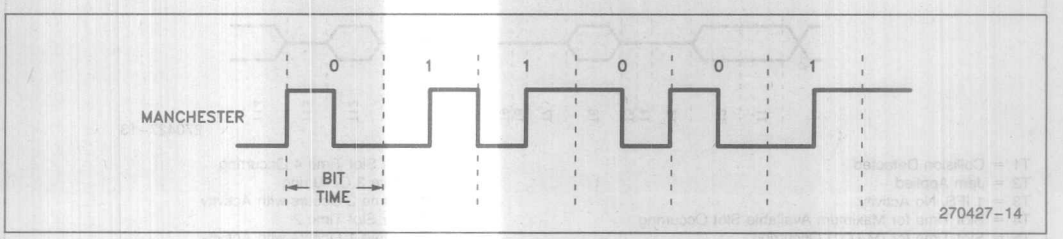


Figure 3.5. Manchester Encoding

3.3.2 SDLC Frame Format

The format of an SDLC frame is shown in Figure 3.6. The frame consists of a Beginning of Frame flag, Address field, Control Field, Information field (optional), a CRC, and the End of Frame flag.



Figure 3.6. Typical SDLC Frame

BOF - The begin of frame flag for SDLC is 01111110. It is only one of two possible combinations that have six consecutive ones in SDLC. The other possibility is an abort character which consists of eight or more consecutive ones. This is because SDLC utilizes a process called bit stuffing. Bit stuffing is the insertion of a 0 as the next bit every time a sequence of five consecutive 1s is detected. The receiver automatically removes a 0 after every consecutive group of five ones. This removal of the 0 bit is referred to as bit stripping. Bit stuffing is discussed in Section 3.3.4. All the procedures required for bit stuffing and bit stripping are automatically handled by the GSC.

In standard SDLC protocol the BOF signals the start of a frame and is limited to 8 bits in length. Since there is no preamble in SDLC the BOF is considered an entire separate field and marks the beginning of the frame. The BOF also serves as the clock synchronization mechanism and the reference point for determining the position of the address and control fields.

ADDRESS - The address field is used to identify which stations the message is intended for. Each secondary station must have a unique address. The primary station must then be made aware of which addresses are assigned to each station. The address length is specified as 8-bits in standard SDLC protocols but it is expandable to 16-bits in the C152. User software can further expand the number of address bits, but the automatic address recognition feature works on a maximum of 16-bits.

In SDLC the addresses are normally unique for each station. However, there are several classes of messages that are intended for more than one station. These messages are called broadcast and group addressed frames. An address consisting of all 1s will always be automatically received by the GSC, this is defined as the broadcast address in SDLC. A group address is an address that is common to more than one station. The GSC provides address masking bits to provide the capability of receiving group addresses.

If desired, the user software can mask off all the bits of the address. This type of masking puts the GSC in a promiscuous mode so that all addresses are received.

CONTROL - The control field is used for initialization of the system, identifying the sequence of a frame, to identify if the message is complete, to tell secondary stations if a response is expected, and acknowledgement of previously sent frames. The user software is responsible for insertion of the control field as the GSC hardware has no provisions for the management of this field. The interpretation and formation of the control field must also be handled by user software. The information following the control field is typically used for information transfer, error reporting, and various other functions. These functions are accomplished by the format of the control field. There are three formats available. The types of formats are Informational, Supervisory, or Unnumbered. Figure 3.7 shows the various format types and how to identify them.

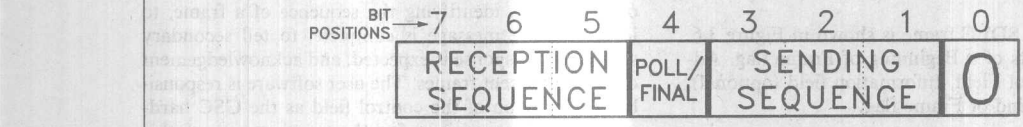
Since the user software is responsible for the implementation of the control field, what follows is a simple explanation on the control field and its functions. For a complete understanding and proper implementation of SDLC, the user should refer to the IBM document, GA27-3093-2, IBM Synchronous Data Link Control General Information. Within that document, is another list of IBM documents which go into detail on the SDLC protocol and its use.

The control field is eight bits wide and the format is determined by bits 0 and 1. If bit 0 is a zero, then the frame is an informational frame. If bit 0 is a one and bit 1 a zero, then it is a supervisory frame, and if bit 0 is a one and bit 1 a one then the frame is an unnumbered frame.

In an informational frame bits 3,2,1 contain the sequence count of the frame being sent.

Bit 4 is the P/F (Poll/Final) bit. If bit 4 equals 1 and originates from the primary, then the secondary station is expected to initiate a transmission. If bit 4 equals 1 and originates from a secondary station, then the frame is the final frame in a transmission.

Bits 7,6,5 contain the sequence count a station expects on the next transmission to it. The sequence count can vary from 000B to 111B. The count then starts over again at 000B after the value 111B is incremented. The acknowledgement is recognized by the receiving station when it decodes bits 7,6,5 of an incoming frame. The station sending the transmission is acknowledging the frames received up to the count represented in bits 7,6,5 (sequence count-1). With this method, up to seven sequential frames may be transmitted prior to an acknowledgement being received. If eight frames were allowed to pass before an acknowledgement, the sequence count would roll over and this would negate the purpose of the sequence numbers.



270427-15

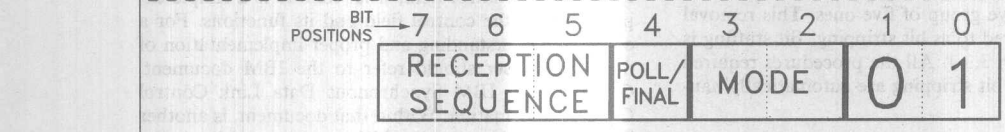
RECEPTION SEQUENCE - The sequence expected in the SENDING SEQUENCE portion of the control byte in the next received frame. This also confirms correct reception of up to seven frames prior to the sequence given.

POLL/FINAL - Identifies the frame as being a polling request from the master station or the last in a series of frames from the master or secondary.

SENDING SEQUENCE - Identifies the sequence of the frame being transmitted.

0 - If bit 0 = 0 the frame is identified as a informational format type.

INFORMATION FORMAT



270427-16

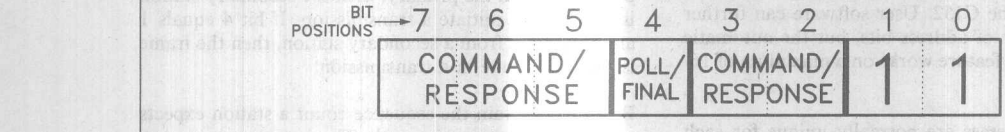
RECEPTION SEQUENCE - Expected sequence of frame for next reception.

POLL/FINAL - Identifies frame as being a polling request from the master station or the last in a series of frames from the master or secondary.

MODE - Identifies whether receiver is ready (00), not ready (10) or a frame was rejected (01). The rejected frame is identified by the reception sequence.

0,1 - If bits 1,0 = 0,1 the frame is identified as a supervisory format type.

SUPERVISORY FORMAT



270427-17

COMMAND/RESPONSE - Identifies the type of command or response.

POLL/FINAL - Identifies frame as being a polling request from the master station or the last in a series of frames from the master or secondary.

1,1 - If bits 1,0 = 1,1 the frame is identified as an unnumbered format type.

NONSEQUENCED FORMAT

270427-18

Figure 3.7. SDLC Control Field

Following the informational control field comes the information to be transferred.

In the supervisory format (bits 1,0 = 0,1) bits 3,2 determine which mode is being used.

When the mode is 00 it indicates that the receive line of the station that sent the supervisory frame is enabled and ready to accept frames.

When the mode is 01, it indicates that previously a received frame was rejected. The value in the receive count identifies which frame(s) need to be retransmitted.

When the mode is 10, the sending station is indicating that its receiver is not ready to accept frames.

Mode 11 is an illegal mode in SDLC protocol.

Bits 7,6,5 represent the value of the sequence the station expects when the next transfer occurs for that station. There is no information following the control field when the supervisory format is used.

In the unnumbered format (bits 1,0 = 1,1) bits 7, 6, 5, 3, 2 (notice bit 4 is missing) indicate commands from the primary to secondary stations or requests of secondary stations to the primary.

The standard commands are:

BITS 7	6	5	3	2	Command
0	0	0	0	0	Unnumbered Information (UI)
0	0	0	0	1	Set initialization mode (SIM)
0	1	0	0	0	Disconnect (DISC)
0	0	1	0	0	Response optional (UP)
1	1	0	0	1	Function descriptor in information field (CFGR)
1	0	1	1	1	Identification in information field. (XID)
1	1	1	0	0	Test pattern in information field. (TEST)

The standard responses are:

BITS 7	6	5	3	2	Command
0	0	0	0	0	Unnumbered information (UI)
0	0	0	0	1	Request for initialization (RIM)
0	0	0	1	1	Station in disconnected mode (DM)
1	0	0	0	1	Invalid frame received (FRMR)
0	1	1	0	0	Unnumbered acknowledgement (UA)
1	1	1	1	1	Signal loss of input (BCN)
1	1	0	0	1	Function descriptor in information field (CFGR)
0	1	0	0	0	Station wants to disconnect (RD)
1	0	1	1	1	Identification in information field (XID)
1	1	1	0	0	Test pattern in information field (TEST)

In an unnumbered frame, information of variable length may follow the control field if UI is used, or information of fixed length may follow if FRMR is used.

As stated earlier, the user software is responsible for the proper management of the control field. This portion of the frame is passed to or from the GSC FIFOs as basic informational type data.

INFO - This is the information field and contains the data that one device on the link wishes to transmit to another device. It can be of any length the user wishes, but must be a multiple of 8 bits. It is possible that some frames may contain no information field. The information field is identified to the receiving stations by the preceding control field and the following CRC. The GSC determines where the last of the information field is by passing the bits through the CRC generator. When the last bit or EOF is received the bits that remain constitute the CRC.

CRC - The Cyclic Redundancy Check (CRC) is an error checking sequence commonly used in serial communications. The C152 offers two types of CRC algorithms, a 16-bit and a 32-bit. The 32-bit algorithm is normally used in CSMA/CD applications and is described in section 3.2.2. In most SDLC applications a 16-bit CRC is used and the hardware configuration that supports 16-bit CRC is shown in Figure 3.8. The generating polynomial that the CRC generator uses with the 16-bit CRC is:

$G(X) = X^{**}16 + X^{**}12 + X^{**}5 + 1$

The way the CRC operates is that as a bit is received it is XOR'd with bit 15 of the current CRC and placed in temporary storage. The result of XOR'ing bit 15 with the received bit is then XOR'd with bit 4 and bit 11 as the CRC is shifted one position to the right. The bit in temporary storage is shifted into position 0.

The required CRC length for SDLC is 16 bits. The CRC is automatically stripped from the frame and not passed on to the CPU. The last 16 bits are then run though the CRC generator to insure that the correct remainder is left. The remainder that is checked for is 001110100001111B (1D0F Hex). If there is a mismatch, an error is generated. The user software has the option of enabling this interrupt so the CPU is notified.

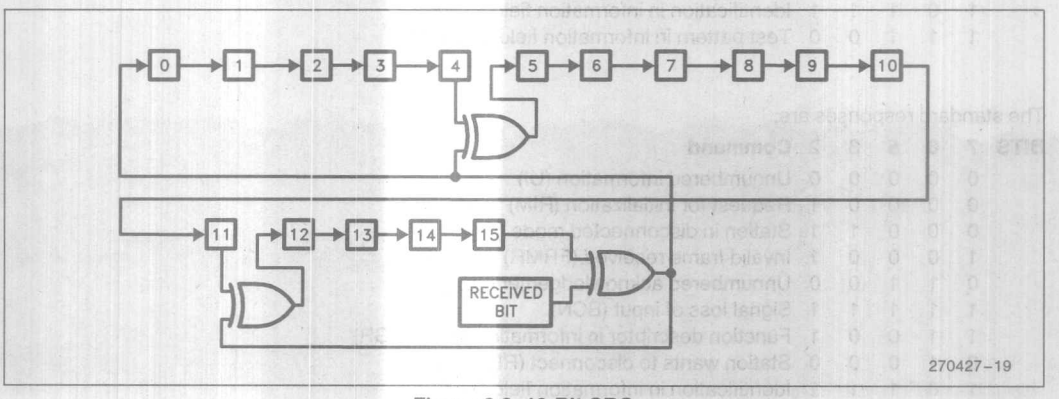


Figure 3.8. 16-Bit CRC

EOF - The End Of Frame (EOF) indicates when the transmission is complete. The EOF is identified by the end flag. An end flag consists of the bit pattern 01111110. The EOF can also serve as the BOF for the next frame.

3.3.3 DATA ENCODING

The transmission of data in SDLC mode is done via NRZI encoding as shown in Figure 3.9. NRZI encoding transmits data by changing the state of the output whenever a 0 is being transmitted. Whenever a 1 is transmitted the state of the output remains the same as the previous bit and remains valid for the entire bit time. When SDLC mode is selected it automatically enables the NRZI encoding on the transmit line and NRZI decoding on the receive line.

3.3.4 BIT STUFFING/STRIPPING

In SDLC mode one of the primary rules of the protocol is that in any normal data transmission, there will never be an occurrence of more than 5 consecutive 1s. The GSC takes care of this housekeeping chore by automatically inserting a 0 after every occurrence of 5 consecutive 1s and the receiver automatically removes a zero after receiving 5 consecutive 1s. All the necessary steps required for implementing bit stuffing and stripping are incorporated into the GSC hardware. This makes the operation transparent to the user. About the only time this operation becomes apparent to the user, is if the actual data on the transmission medium is being monitored by a device that is not aware of the automatic insertion of 0s. The bit stuffing/stripping guarantees that there will be at least one transition every 6 bit times while the line is active.

3.3.5 SENDING ABORT CHARACTER

An abort character is one of the exceptions to the rule that disallows more than 5 consecutive 1s. The abort character consists of any occurrence of seven or more consecutive ones. The simplest way for the C152 to send an abort character is to clear the TEN bit. This causes the output to be disabled which, in turn, forces it to a constant high state. The delay necessary to insure that the link is high for seven bit times, is a task that needs to be handled by user software. Other methods of sending an abort character are using the IFS register or using the Raw Transmit mode. Using IFS still entails clearing the TEN bit, but TEN can be immediately re-enabled. The next message will not begin until the IFS expires. The IFS begins timing out as soon as $\overline{\text{DEN}}$ goes high which identifies the end of transmission. This also requires that IFS contain a value equal to or greater than 8. This method may have the undesirable effect that $\overline{\text{DEN}}$ goes high and disables the external drivers. The other alternative is to switch to Raw Transmit mode. Then, writing 0FFH to TFIFO would generate a high output for 8 bit times. This method would leave $\overline{\text{DEN}}$ active during the transmission of the abort character.

When the receiver detects seven or more consecutive 1s and data has been loaded into the receive FIFO, the RCABT flag is set in RSTAT and that frame is ignored. If no data has been loaded into the receive FIFO, there are no abort flags set and that frame is just ignored. A retransmitted frame may immediately follow an abort character, provided the proper flags are used.

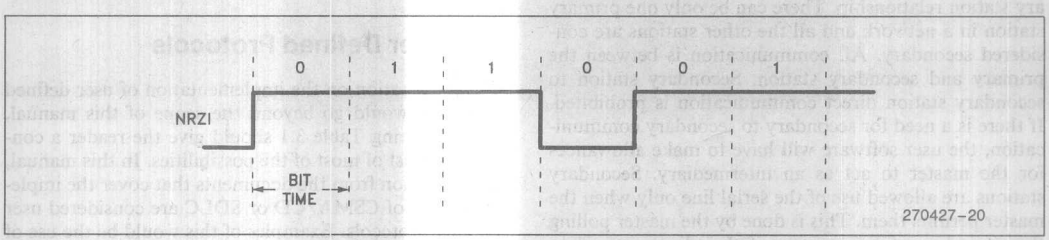


Figure 3.9. NRZI Encoding

3.3.6 LINE IDLE

If 15 or more consecutive 1s are detected by the receiver the Line Idle bit (LNI) in TSTAT is set. The seven 1s from the abort character may be included when sensing for a line idle condition. The same methods used for sending the Abort character can be used for creating the Idle condition. However, the values would need to be changed to reflect 15 bit times, instead of seven bit times.

3.3.7 ACKNOWLEDGEMENT

Acknowledgment in SDLC is an implied acknowledge and is contained in the control field. Part of the control frame is the sequence number of the next expected frame. This sequence number is called the Receive Count. In transmitting the Receive Count, the receiver is in fact acknowledging all the previous frames prior to the count that was transmitted. This allows for the transmission of up to seven frames before an acknowledge is required back to the transmitter. The limitation of seven frames is necessary because the Receive Count in the control field is limited to three binary digits. This means that if an eighth transmission occurred this would cause the next Receive Count to repeat the first count that still is waiting for an acknowledge. This would defeat the purpose of the acknowledgement. The processing and general maintenance of the sequence count must be done by the user software. The Hardware Based Acknowledge option that is provided in the C152 is not compatible with standard SDLC protocol.

3.3.8 PRIMARY/SECONDARY STATIONS

All SDLC networks are based upon a primary/secondary station relationship. There can be only one primary station in a network and all the other stations are considered secondary. All communication is between the primary and secondary station. Secondary station to secondary station direct communication is prohibited. If there is a need for secondary to secondary communication, the user software will have to make allowances for the master to act as an intermediary. Secondary stations are allowed use of the serial line only when the master permits them. This is done by the master polling the secondary stations to see if they have a need to access the serial line. This should prevent any collisions from occurring, provided each secondary station has its own unique address. This arrangement also partially determines the types of networks supported. Normal SDLC networks consist of point-to-point, multi-drop, or ring configurations and the C152 supports all of these. However, some SDLC processors support an automatic one bit delay at each node that is not supported by the C152. In a "Loop Mode" configuration, is is necessary that the transmission be delayed from the reception of the frames from the upstream station before passing the message to the downstream station. This delay is necessary so that a station can decode its own address before the message is passed on. The various networks are shown in Figure 3.10.

3.3.9 HDLC/SDLC COMPARISON

HDLC (High level Data Link Control) is a standard adopted by the International Standards Organization (ISO). The HDLC standard is defined in the ISO document #ISO 6159 - HDLC unbalanced classes of procedures. IBM developed the SDLC protocol as a subset of HDLC. SDLC conforms to HDLC protocol requirements, but is more restrictive. SDLC contains a more precise definition on the modes of operation.

Some of the major differences between SDLC and HDLC are:

SDLC	HDLC
Unbalanced (primary/ secondary)	Balanced (peer to peer)
Modulo 8 (no extensions allowed, up to 7 outstanding frames before acknowledge is required)	Modulo 128 (up to 127 outstanding frames before acknowledge is required)
8-bit addressing only	Extended addressing
Byte aligned data	Variable size of data

The C152 does not support HDLC implementation requiring data alignment other than byte alignment. The user will find that many of the protocol parameters are programmable in the C152 which allows easy implementation of proprietary or standard HDLC network. User software needs to implement the control field functions.

3.4 User Defined Protocols

The explanation on the implementation of user defined protocols would go beyond the scope of this manual, but examining Table 3.1 should give the reader a consolidated list of most of the possibilities. In this manual, any deviation from the documents that cover the implementation of CSMA/CD or SDLC are considered user defined protocols. Examples of this would be the use of SDLC with the 32-bit CRC selected or CSMA/CD with hardware based acknowledge.

3.5 Using the GSC

3.5.1 LINE DISCIPLINE

Line discipline is how the management of the transfer of data over the physical medium is controlled. Two types of line discipline will be discussed in this section: full duplex and half duplex.

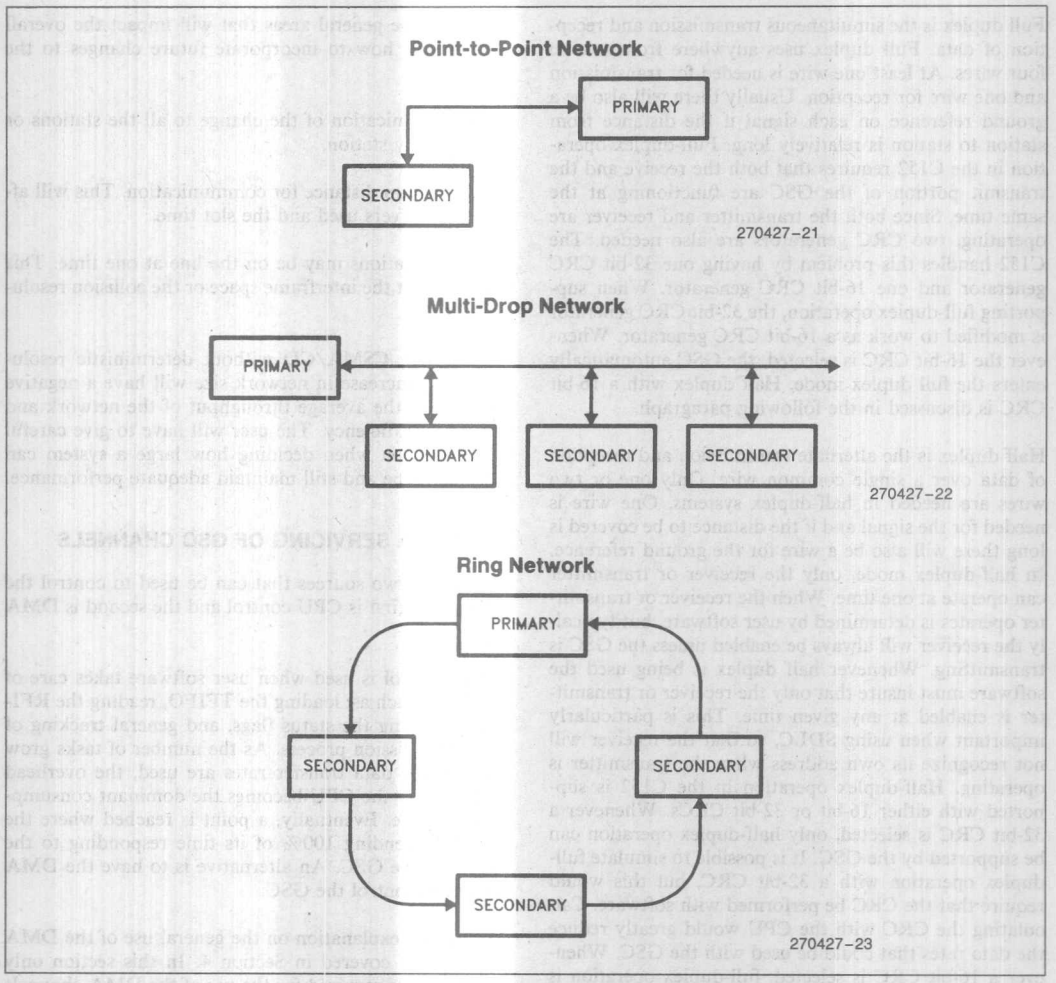


Figure 3.10. SDLC Networks

Full duplex is the simultaneous transmission and reception of data. Full duplex uses anywhere from two to four wires. At least one wire is needed for transmission and one wire for reception. Usually there will also be a ground reference on each signal if the distance from station to station is relatively long. Full-duplex operation in the C152 requires that both the receive and the transmit portion of the GSC are functioning at the same time. Since both the transmitter and receiver are operating, two CRC generators are also needed. The C152 handles this problem by having one 32-bit CRC generator and one 16-bit CRC generator. When supporting full-duplex operation, the 32-bit CRC generator is modified to work as a 16-bit CRC generator. Whenever the 16-bit CRC is selected, the GSC automatically enters the full duplex mode. Half duplex with a 16-bit CRC is discussed in the following paragraph.

Half duplex is the alternate transmission and reception of data over a single common wire. Only one or two wires are needed in half-duplex systems. One wire is needed for the signal and if the distance to be covered is long there will also be a wire for the ground reference. In half-duplex mode, only the receiver or transmitter can operate at one time. When the receiver or transmitter operates is determined by user software, but typically the receiver will always be enabled unless the GSC is transmitting. Whenever half duplex is being used the software must insure that only the receiver or transmitter is enabled at any given time. This is particularly important when using SDLC, so that the receiver will not recognize its own address when the transmitter is operating. Half-duplex operation in the C152 is supported with either 16-bit or 32-bit CRCs. Whenever a 32-bit CRC is selected, only half-duplex operation can be supported by the GSC. It is possible to simulate full-duplex operation with a 32-bit CRC, but this would require that the CRC be performed with software. Calculating the CRC with the CPU would greatly reduce the data rates that could be used with the GSC. Whenever a 16-bit CRC is selected, full-duplex operation is automatically chosen and the GSC must be reconfigured if half-duplex operation is preferred.

3.5.2 PLANNING FOR NETWORK CHANGES AND EXPANSIONS

A complete explanation on how to plan for network expansion will not be covered in this manual as there are far too many possibilities that would need to be discussed. But there are several areas that will have major impact when allowing for changes in the system. In cases where there will never be any changes allowed, expansion plans become a mute issue. However, it is strongly suggested that there always be some allowance for future modifications.

Some of the general areas that will impact the overall scheme on how to incorporate future changes to the system are:

1) Communication of the change to all the stations or the primary station.

2) Maximum distance for communication. This will affect the drivers used and the slot time.

3) More stations may be on the line at one time. This may impact the interframe space or the collision resolution used.

4) If using CSMA/CD without deterministic resolution, any increase in network size will have a negative impact on the average throughput of the network and lower the efficiency. The user will have to give careful consideration when deciding how large a system can ultimately be and still maintain adequate performance.

3.5.3 DMA SERVICING OF GSC CHANNELS

There are two sources that can be used to control the GSC. The first is CPU control and the second is DMA control.

CPU control is used when user software takes care of the tasks such as: loading the TFIFO, reading the RFIFO, checking the status flags, and general tracking of the transmission process. As the number of tasks grow and higher data transfer rates are used, the overhead required by the CPU becomes the dominant consumption of time. Eventually, a point is reached where the CPU is spending 100% of its time responding to the needs of the GSC. An alternative is to have the DMA channels control the GSC.

A detailed explanation on the general use of the DMA channels is covered in Section 4. In this section only those details required for the use of the DMA channels with the GSC will be covered.

The DMA channels can be configured by user software so that the GSC data transfers are serviced by the DMA controller. Since there are two DMA channels, one channel can be used to service the receiver, and one channel can be used to service the transmitter. In using the DMA channels, the CPU is relieved of much of the time required to do the basic servicing of the GSC buffers. The types of servicing that the DMA channels can provide are: loading of the transmit FIFO, removing data from the receive FIFO, notification of the CPU when the transmission or reception has ended, and response to certain error conditions. When using the

DMA channels the source or destination of the data intended for serial transmission can be internal data memory, external data memory, or any of the SFRs.

The only tasks required after initialization of the DMA and GSC registers are enabling the proper interrupts and informing the DMA controller when to start. After the DMA channels are started all that is required of the CPU is to respond to error conditions or wait until the end of transmission.

Initialization of the DMA channels requires setting up the control, source, and destination address registers. On the DMA channel servicing the receiver, the control register needs to be loaded as follows: DCONn.2 = 0, this sets the transfer mode so that response is to GSC interrupts and put the DMA control in alternate cycle mode; DCONn.3 = 1, this enables the demand mode; DCONn.4 = 0, this clears the automatic increment option for the source address; and DCONn.5 = 1, this defines the source as SFR. The DMA channel servicing the receiver also needs its source address register to contain the address of RFIFO (SARHN = XXH, SARLN = 0F4H). On the DMA channel servicing the transmitter, the control register needs to be loaded as follows: DCONn.2 = 0; DCONn.3 = 1; DCONn.6 = 0, this clears the automatic increment option for the destination address; and DCONn.7 = 1, this sets the destination as SFR. The DMA channel serving the transmitter also requires that its destination address register contains the address of TFIFO (DARHN = XXH, DARLN = 85H). Assuming that DCON0 would be serving the receiver and DCON1 the transmitter, DCON0 would be loaded with XX1010X0B and DCON1 would be loaded with 10XX10X0B. The contents of SARH0 and DARH1 do not have any impact when using internal SFRs as the source or destination.

When using the DMA channels to service the GSC, the byte count registers will also need to be initialized.

The Done flag for the DMA channel servicing the receiver should be used if fixed packet lengths only are being transmitted or to insure that memory is not overwritten by long received data packets. Overwriting of data can occur when using a smaller buffer than the packet size. In these cases the servicing of the DMA and/or GSC would be in response to the DMA Done flag when the byte count reaches zero.

In some cases the buffer size is not the limiting factor and the packet lengths will be unknown. In these cases it would be desirable to eliminate the function of the Done flag. To effectively disable the Done flag for the DMA channel servicing the receiver, the byte count should be set to some number larger than any packet that will be received, up to 64K. If not using the Done flag, then GSC servicing would be driven by the receive Done (RDN) flag and/or interrupt. RDN is set when the EOF is detected. When using the RDN flag, RFNE should also be checked to insure that all the data has been emptied out of the receive FIFO.

The byte count register is used for all transmissions and this means that all packets going out will have to be of the same length or the length of the packet to be sent will have to be known prior to the start of transmission. When using the DMA channels to service the GSC transmitter, there is no practical way to disable the Done flag. This is because the transmit done flag (TDN) is set when the transmit FIFO is empty and the last message bit has been transmitted. But, when using the DMA channel to service the transmitter, loads to the TFIFO continue to occur until the byte count reaches 0. This makes it impossible to use TDN as a flag to stop the DMA transfers to TFIFO. It is possible to examine some other registers or conditions, such as the current byte count, to determine when to stop the DMA transfers to TFIFO, but this is not recommended as a way to service the DMA and GSC when transmitting because frequent reading of the DMA registers will cause the effective DMA transfer rate to slow down.

When using the DMA channels, initialization of the GSC would be exactly the same as normal except that TSTAT.0 = 1 (DMA), this informs the GSC that the DMA channels are going to be used to service the GSC. Although only TSTAT is written to, both the receiver and transmitter use this same DMA bit.

The interrupts EGSTE (IEN1.5), GSC transmit error; EGSTV (IEN1.3), GSC transmit valid; EGSRE (IEN1.1), GSC receive error; and EGSRV (IEN1.0), GSC receive valid; need to be enabled. The DMA interrupts are normally not used when servicing the GSC with the DMA channels. To ensure that the DMA interrupts are not responded to is a function of the user software and should be checked by the software to make sure they are not enabled. Priority for these interrupts can also be set at this time. Whether to use high or low priority needs to be decided by the user. When responding to the GSC interrupts, if a buffer is being used to store the GSC information, then the DMA registers used for the buffer will probably need updating.

After this initialization, all that needs to be done when the GSC is actually going to be used is: load the byte count, set-up the source addresses for the DMA channel servicing the transmitter, set-up the destination addresses for the DMA channel servicing the receiver, and start the DMA transfer. The GSC enable bits should be set first and then the GO bits for the DMA. This initiates the data transfers.

This simplifies the maintenance of the GSC and can make the implementation of an external buffer for packetized information automatic.

An external buffer can be used as the source of data for transmission, or the destination of data from the receiver. In this arrangement, the message size is limited to the RAM size or 64K, whichever is smaller. By using an external buffer, the data can be accessed by other devices which may want access to the serial data. The amount of time required for the external data moves will also decrease. Under CPU control, a "MOVX" command would take 24 oscillator periods to complete. Under DMA control, external to internal, or internal to external, data moves take only 12 oscillator periods.

3.5.4 BAUD RATE

The GSC baud rate is determined by the contents of the SFR, BAUD, or the external clock. The formula used to determine the baud rate when using the internal clock is:

(fosc)/((BAUD+1)*8)

For example if a 12 MHz oscillator is used the baud rate can vary from:

12,000,000/((0+1)*8) = 1.5 MBPS

to:

12,000,000/((255+1)*8) = 5.859 KBPS

There are certain requirements that the external clock will need to meet. These requirements are specified in the data sheet. For a description of the use of the GSC with external clock please read Section 3.5.11.

3.5.5 INITIALIZATION

Initialization can be broken down into two major components, 1) initialization of the component so that its serial port is capable of proper communication; and 2) initialization of the system or a station so that intelligible communication can take place.

Most of the initialization of the component has already been discussed in the previous sections. Those items not covered are the parameters required for the component to effectively communicate with other components. These types of issues are common to both system and component initialization and will be covered in the following text.

Initialization of the system can be broken down into several steps. First, are the assumptions of each network station.

The first assumption is that the type of data encoding to be used is predetermined for the system and that each station will adhere to the same basic rules defining that encoding. The second assumption is that the basic protocol and line discipline is predetermined and known. This means that all stations are using CSMA/CD or SDLC or whatever, and that all stations are either full or half duplex. The third assumption is that the baud rate is preset for the whole system. Although the baud rate could probably be determined by the microprocessor just by monitoring the link, it will make it much simpler if the baud rate is known in advance.

One of the first things that will be required during system initialization is the assignment of unique addresses for each station. In a two-station only environment this is not necessary and can be ignored. However, keep in mind, that all systems should be constructed for easy future expansions. Therefore, even in only a two station system, addresses should be assigned. There are three basic ways in which addresses can be assigned. The first, and most common is preassigned addresses that are loaded into the station by the user. This could be done with a DIP-switch, through a keyboard. The second method of assigning addresses is to randomly assign an address and then check for its uniqueness throughout the system, and the third method is to make an inquiry to the system for the assignment of a unique address. Once the method of address assignment is determined, the method should become part of the specifications for the system to which all additions will have to adhere. This, then, is the final assumption.

The negotiation process may not be clear for some readers. The following two procedures are given as a guideline for dynamic address assignment.

In the first procedure, a station assumes a random address and then checks for its uniqueness throughout the system. As a station is initialized into the system it sends out a message containing its assumed address. The format of the message should be such that any station decoding the address recognizes it as a request for initialization. If that address is already used, the receiving station returns a message, with its own address stating that the address in question is already taken. The initializing station then picks another address. When the initializing station sends its inquiry for the address check, a timer is also started. If the timer expires before the inquiry is responded to, then that station assumes the address chosen is okay.

In the second procedure, an initializing station asks for an address assignment from the system. This requires that some station on the link take care of the task of maintaining a record of which addresses are used. This station will be called station-1. When the initializing station, called station-2, gets on the link, it sends out a message with a broadcast address. The format of the message should be such that all other stations on the link recognize it as a request for address assignment. Part of the message from station-2 is a random number generated by the station requesting the address. Station-2 then examines all received messages for this random number. The random number could be the address of the received message or could be within the information section of a broadcast frame. All the stations, except station-1, on the link should ignore the initialization request. Station-1, upon receiving the initialization request, assigns an address and returns it to station-2. Station-1 will be required to format the message in such a manner so that all stations on the link recognize it as a response to initialization. This means that all stations except station-2 ignore the return message.

3.5.6 TEST MODES

There are two test modes associated with the GSC that are made available to the user. The test modes are named Raw Receive and Raw Transmit. The test modes are selected by the proper setting of the two mode bits in GMOD (M0 = GMOD.5, M1 = GMOD.6). If M1,M0 = 0,1 then Raw Transmit is selected. If M1,M0 = 1,0 then Raw Receive is enabled.

In Raw Transmit, the transmit output is internally connected to the Receiver input. This is intended to be used as a local loop-back test mode, so that all data written to the transmitter will be returned by the receiver. Raw Transmit can also be used to transmit user data. If Raw Transmit is used in this way the data is emitted with no preamble, flag, address, CRC, and no bit insertion. The data is still encoded with whatever format is selected, Manchester with CSMA/CD, NRZI with SDLC or as NRZ if external clocks are used. The receiver still operates as normal and in this mode most of the receive functions can be tested.

In Raw Receive, the transmitter should be externally connected to the receiver. To do this a port pin should be used to enable an external device to connect the two pins together. In Raw Receive mode the receiver acts as normal except that all bytes following the BOF are loaded into the receive FIFO, including the CRC. Also address recognition is not active but needs to be performed in software. If SDLC is selected as the protocol, zero-bit deletion is still enabled. The transmitter still operates as normal and in this mode most of the transmitter functions and an external transceiver can be tested. This is also the only way that the CRC can be read by the CPU, but the CRC error bit will not be set.

3.5.7 EXTERNAL DRIVER INTERFACE

A signal is provided from the C152 to enable transmitter drivers for the serial link. This is provided for systems that require more than what the GSC ports are capable of delivering. The voltage and currents that the GSC is capable of providing are the same levels as those for normal port operation. The signal used to enable the external drivers is $\overline{DEN}$. No similar signal is needed for the receiver.

3.5.8 JITTER (RECEIVE)

Data jitter is the difference between the actual transmitted waveform and the exact calculated value(s). In NRZI, data jitter would be how much the actual waveform exceeds or falls short of one calculated bit time. A bit time equals 1/baud rate. If using Manchester encoding, there can be two transitions during one bit time as shown in Figure 3.11. This causes a second parameter to be considered when trying to figure out the complete data jitter amount. This other parameter is the half-bit jitter. The half-bit jitter is comprised of the difference in time that the half-bit transition actually occurs and the calculated value. Jitter is important because if the transition occurs too soon it is considered noise, and if the transition occurs too late, then either the bit is missed or a collision is assumed.

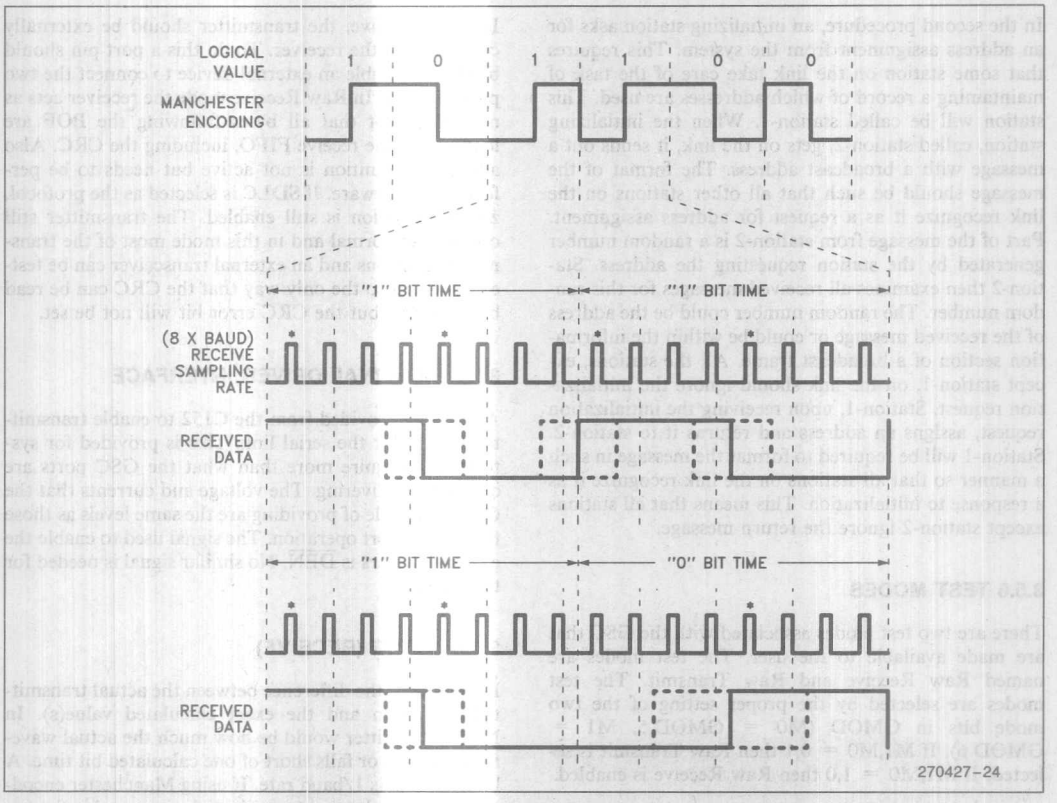


Figure 3.11. Jitter

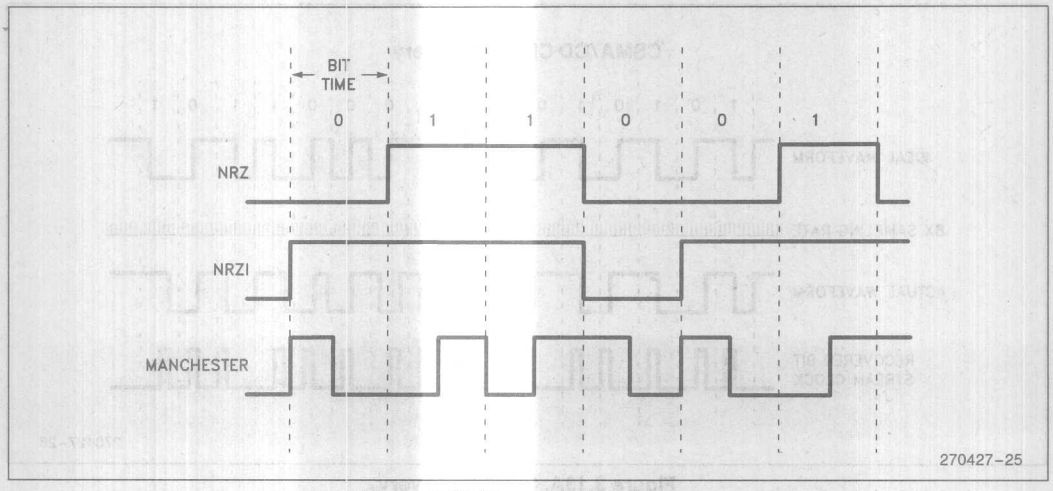


Figure 3.12. Transmit Waveforms

3.5.9 Transmit Waveforms

The GSC is capable of three types of data encoding, Manchester, NRZI, and NRZ. Figure 3.12 shows examples of all three types of data encoding.

3.5.10 Receiver Clock Recovery

The receiver is always monitored at eight times the baud rate frequency, except when an external clock is used. When using an external clock the receiver is loaded during the clock cycle.

In CSMA/CD mode the receiver synchronizes to the transmitted data during the preamble. If a pulse is detected as being too short it is assumed to be noise or a collision. If a pulse is too long it is assumed to be a collision or an idle condition.

In SDLC the synchronization takes place during the BOF flag. In addition, pulses less than four sample periods are ignored, and assumed to be noise. This sets a lower limit on the pulse size of received zeros.

In CSMA/CD the preamble consists of alternating 1s and 0s. Consequently, the preamble looks like the waveform in Figure 3.13A and 3.13B.

3.5.11 External Clocking

To select external clocking, the user is given three choices. External clocking can be used with the transmitter, with the receiver, or with both. To select external clocking for the transmitter, XTCLK (GMOD.7) has to be set to a 1. To select external clocking for the receiver, XRCLK (PCON.3) has to be set to a 1. Setting both bits to 1 forces external clocking for the receiver and transmitter.

The external transmit clock is applied to pin 4 ($\overline{TXC}$), P1.3. The external receive clock is applied to pin 5 ($\overline{RXC}$), P1.4. To enable the external clock function on the port pin, that pin has to be set to a 1 in the appropriate SFR, P1.

Whenever the external clock option is used, the format of the transmitted and received data is restricted to NRZ encoding and the protocol is restricted to SDLC. With external clock, the bit stuffing/stripping is still active with SDLC protocol.

3.6 GSC Operation

3.6.1 Determining Line Discipline

In normal operation the GSC uses full or half duplex operation. When using a 32-bit CRC (GMOD.3 = 1), operation can only be half duplex. If using a 16-bit CRC (GMOD.3 = 0), full duplex is selected by default. When using a 16-bit CRC the receiver can be turned off while transmitting (RSTAT.1 = 0), and the transmitter can be turned off during reception (TSTAT.1 = 0). This simulates half-duplex operation when using a 16-bit CRC.

Normally, HDLC uses a 16-bit CRC, so half duplex is determined by turning off the receiver or transmitter. This is so that the receiver will not detect its own ad-

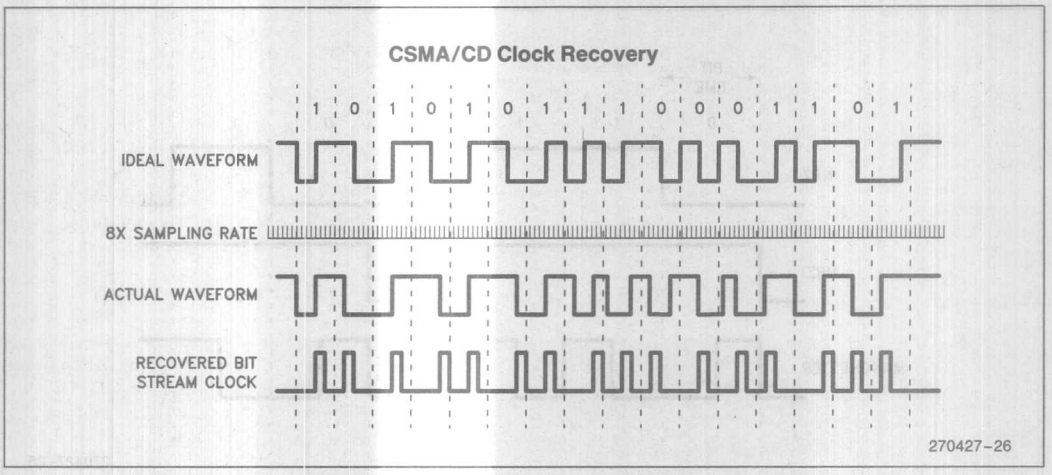


Figure 3.13A. Clock Recovery

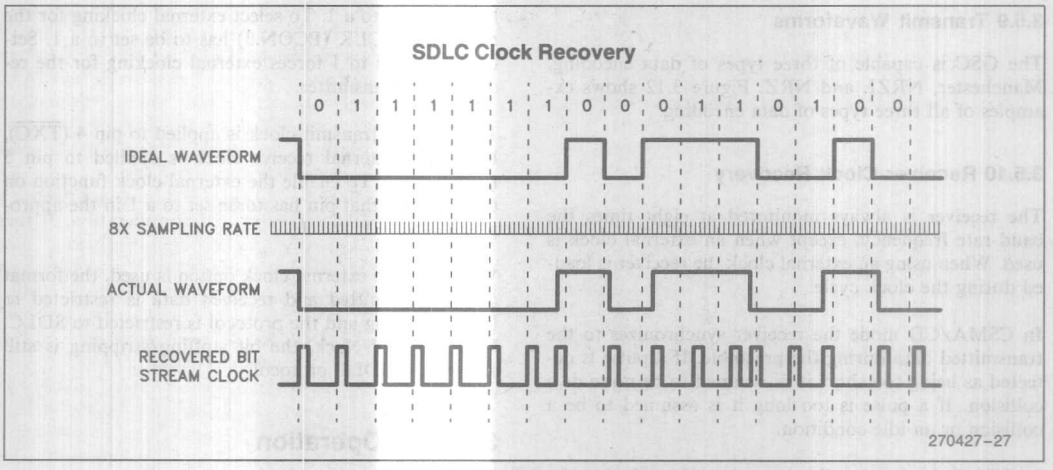


Figure 3.13B. Clock Recovery

dress as transmission takes place. This also needs to be done when using CSMA/CD with a 16-bit CRC for the same reason.

3.6.2 CPU/DMA CONTROL OF THE GSC

The data for transmission or reception can be handled by either the CPU (TSTAT.0 = 0) or DMA controller (TSTAT.0 = 1). This allows the user two sets of flags to control the FIFO. Associated with these flags are interrupts, which may be enabled by the user software. Either one or both sets of flags may be used at the same time.

In CPU control mode the flags (RFNE,TFNF) are generated by the condition of the receive or transmit FIFO's. After loading a byte into the transmit FIFO, there is a one machine cycle latency until the TFNF flag is updated. Because of this latency, the status of TFNF should not be checked immediately following the instruction to load the transmit FIFO. If using the interrupts to service the transmit FIFO,the one machine cycle of latency must be considered if the TFNF flag is checked prior to leaving the subroutine.

When using the CPU for control, transmission normally is initiated by setting the TEN bit (TSTAT.1) and then writing to TFIFO. TEN must be set before loading the transmit FIFO, as setting TEN clears the transmit FIFO. TCDCNT should also be checked by user software and cleared if a collision occurred on a prior transmission.

To enable the receiver, GREN (RSTAT.1) is set. After GREN is set, the GSC begins to look for a valid BOF. After detecting a valid BOF the GSC attempts to match the received address byte(s) against the address match registers. When a match occurs the frame is loaded into the GSC. Due to the CRC strip hardware, there is a 40 or 24 bit time delay following the BOF until the first data byte is loaded into RFIFO if the 32 or 16 bit CRC is chosen. If the end of frame is detected before data is loaded into the receive FIFO, the receiver ignores that frame.

If the receiver detects a collision during reception in CSMA/CD mode and if any bytes have been loaded into the receive FIFO, the RCABT flag is set. The GSC hardware then halts reception and resets GREN. The user software needs to filter any collision fragment data which may have been received. If the collision occurred prior to the data being loaded into RFIFO the CPU is not notified and the receiver is left enabled. At the end of a reception the RDN bit is set and GREN is cleared. In HABEN mode this causes an acknowledgement to be transmitted if the frame did not have a broadcast or multi-cast address. The user software can enable the interrupt for RDN to determine when a frame is completed.

In DMA mode the interrupts are generated by the internal "transmit/receive done" (TDN,RDN) conditions. When the CPU responds to TDN or RDN, checks are performed to see if the transmit underrun error has occurred. The underrun condition is only checked when using the DMA channels.

Upon power up the CPU mode is initialized. General DMA control is covered in Section 4.0. DMA control of the GSC is covered in Section 3.5.4. If DMA is to be used for serving the GSC, it must be configured into the serial channel demand mode and the DMA bit in TSTAT has to be set.

3.6.3 COLLISIONS AND BACKOFF

The actions that are taken by the GSC if a collision occurs while transmitting depend on where the collision occurs. If a collision occurs in CSMA/CD mode following the preamble and BOF flag, the TCDT flag is set and the transmit hardware completes a jam. When this type of collision occurs, there will be no automatic retry at transmission. After the jam, control is returned to the CPU and user software must then initiate whatever actions are necessary for a proper recovery. The possibility that data might have been loaded into or from the GSC deserves special consideration. If these fragments of a message have been passed on to other devices, user software may have to perform some extensive error handling or notification. Before starting a new message, the transmit and receive FIFOs will need to be cleared. If DMA servicing is being used the pointers must also be reinitialized. It should be noted that a collision should never occur after the BOF flag in a well designed system, since the system slot time will likely be less than the preamble length. The occurrence of such a situation is normally due to a station on the link that is not adhering to proper CSMA/CD protocol or is not using the same timings as the rest of the network.

A collision occurring during the preamble or BOF flag is the normal type of collision that is expected. When this type of collision occurs the GSC automatically handles the retransmission attempts for as many as eight tries. If on the eighth attempt a collision occurs, the transmitter is disabled, although the jam and backoff are performed. If enabled, the CPU is then interrupted. The user software should then determine what action to take. The possibilities range from just reporting the error and aborting transmission to reinitializing the serial channel registers and attempt retransmission.

If less than eight attempts are desired TCDCNT can be loaded with some value which will reduce the number of collisions possible before TCDCNT overflows. The value loaded should consist of all 1s as the least significant bits, e.g. 7, 0FH, 3FH. A solid block of 1s is suggested because TCDCNT is used as a mask when generating the random slot number assignment. The TCDCNT register operates by shifting the contents one bit position to the left as each collision is detected. As each shift occurs a 1 is loaded into the LSB. When TCDCNT overflows, GSC operation stops and the CPU is notified by the setting of the TCDT bit which can flag an interrupt.

The amount of time that the GSC has before it must be ready to retransmit after a collision is determined by the mode which is selected. The mode is determined M0 (GMOD.5) and M1 (GMOD.6). If M0 and M1 equal 0,0 (normal backoff) then the minimum period before retransmission will be either the interframe space or the backoff period, whichever is longer. If M0 and M1 equal 1,1 (alternate backoff) then the minimum period before retransmission will be the interframe space plus the backoff period. Both of these are shown in Figure 3.4. Alternate backoff must be enabled if using deterministic resolution. If the GSC is not ready to retransmit by the time its assigned slot becomes available, the slot time is lost and the station must wait until the collision resolution time period has passed.

Instead of waiting for the collision resolution to pass, the transmission could be aborted. The decision to abort is usually dependent on the number of stations on the link and how many collisions have already occurred. The number of collisions can be obtained by examining the register, TCDCNT. The abort is normally implemented by clearing TEN. The new transmission begins by setting TEN and loading TFIFO. The minimum amount of time available to initiate a retransmission would be one interframe space period after the line is sensed as being idle.

As the number of stations approach 256 the probability of a successful transmission decreases rapidly. If there are more than 256 stations involved in the collision there would be no resolution since at least two of the stations will always have the same backoff interval selected.

All the stations monitor the link as long as that station is active, even if not attempting to transmit. This is to ensure that each station always defers the minimum amount of time before attempting a transmission and so that addresses are recognized. However, the collision detect circuitry operates slightly differently.

In normal back-off mode, a transmitting station always monitors the link while transmitting. If a collision is detected one or more of the transmitting stations apply the jam signal and all transmitting stations enter the back-off algorithm. The receiving stations also constantly monitor for a collision but do not take part in the resolution phase. This allows a station to try to transmit in the middle of a resolution period. This in turn may or may not cause another collision. If the new station trying to transmit on the link does so during an unused slot time then there will probably not be a collision. If trying to transmit during a used slot time, then there will probably be a collision. The actions the receiver does take when detecting a collision is to just stop receiving data if data has not been loaded into RFIFO or to stop reception, clear receiver enable (REN) and set the receiver abort flag (RCABT - RSTAT.6).

If deterministic resolution is used, the transmitting stations go through pretty much the same process as in normal back-off, except that the slots are predetermined. All the receivers go through the back-off algorithm and may only transmit during their assigned slot.

3.6.4 SUCCESSFUL ENDING OF TRANSMISSIONS AND RECEPTIONS

In both CSMA/CD and SDLC modes, the TDN bit is set and TEN cleared at the end of a successful transmission. The end of the transmission occurs when the TFIFO is empty and the last byte has been transmitted. In CSMA/CD the user should clear the TCDCNT register after successful transmission.

At the end of a successful reception, the RDN bit is set and GREN is cleared. The end of reception occurs when the EOF flag is detected by the GSC hardware.

3.7 Register Descriptions

ADR0,1,2,3 (95H, 0A5H, 0B5H, 0C5H) - Address Match Registers 0,1,2,3 - Contains the address match values which determines which data will be accepted as valid. In 8 bit addressing mode, a match with any of the four registers will trigger acceptance. In 16 bit addressing mode a match with ADR1:ADR0 or ADR3:ADR2 will be accepted. Addressing mode is determined in GMOD (AL).

AMSK0,1 (0D5H, 0E5H) - Address Match Mask 0,1 - Identifies which bits in ADR0,1 are "don't care" bits. Writing a one to a bit in AMSK0,1 masks out that corresponding bit in ADDR0,1.

BAUD (94H) - GSC Baud Rate Generator - Contains the value of the programmable baud rate. The data rate will equal (frequency of the oscillator)/((BAUD + 1) × (8)). Writing to BAUD actually stores the value in a reload register. The reload register contents are copied into the BAUD register when the Baud register decrements to 00H. Reading BAUD yields the current timer value. A read during GSC operation will give a value that may not be current because the timer could decrement between the time it is read by the CPU and by the time the value is loaded into its destination.

BKOFF (0C4H) - Backoff Timer - The backoff timer is an eight bit count-down timer with a clock period equal to one slot time. The backoff time is used in the CSMA/CD collision resolution algorithm. The user software may read the timer but the value may be invalid as the timer is clocked asynchronously to the CPU. Writing to 0C4H will have no effect.

GMOD (84H)

7	6	5	4	3	2	1	0
XTCLK	M1	M0	AL	CT	PL1	PL0	PR

Figure 3.14. GMOD

GMOD.0 (PR) - Protocol - If set, SDLC protocols with NRZI encoding and SDLC flags are used. If cleared, CSMA/CD link access with Manchester encoding is used. The user software is responsible for setting or clearing this flag.

GMOD.1,2 (PL0,1) - Preamble length

PL1	PL0	LENGTH (BITS)
0	0	0
0	1	8
1	0	32
1	1	64

The length includes the two bit Begin Of Frame (BOF) flag in CSMA/CD but does not include the SDLC flag. In SDLC mode, the BOF is an SDLC flag, otherwise it is two consecutive ones. Zero length is not compatible in CSMA/CD mode. The user software is responsible for setting or clearing these bits.

GMOD.3 (CT) - CRC Type - If set, 32 bit AUTODIN-II-32 is used. If cleared, 16 bit CRC-CCITT is used. The user software is responsible for setting or clearing this flag.

GMOD.4 (AL) - Address Length - If set, 16 bit addressing is used. If cleared, 8 bit addressing is used. In 8 bit mode a match with any of the 4 address registers will be accepted (ADR0, ADR1, ADR2, ADR3). "Don't Care" bits may be masked in ADR0 and ADR1 with AMSK0 and AMSK1. In 16 bit mode, addresses are matched against "ADR1:ADR0" or "ADR3:ADR2". Again, "Don't Care" bits in ADR1:ADR0 can be masked in AMSK1:AMSK0. A received address of all ones will always be recognized in any mode. The user software is responsible for setting or clearing this flag.

GMOD.5,6 (M0,M1) - Mode Select - Two test modes, an optional "alternate backoff" mode, or normal backoff can be enabled with these two bits. The user software is responsible for setting or clearing the mode bits.

M1	M0	Mode
0	0	Normal
0	1	Raw Transmit
1	0	Raw Receive
1	1	Alternate Backoff

In raw receive mode, the receiver operates as normal except that all the bytes following the BOF are loaded into the receive FIFO, including the CRC. The transmitter operates as normal.

In raw transmit mode the transmit output is internally connected to the receiver input. The internal connection is not at the actual port pin, but inside the port latch. All data transmitted is done without a preamble, flag or zero bit insertion, and without appending a CRC. The receiver operates as normal. Zero bit deletion is performed.

In alternate backoff mode the standard backoff process is modified so the the backoff is delayed until the end of the IFS. This should help to prevent collisions constantly happening because the IFS time is usually larger than the slot time.

GMOD.7 (XTCLK) - External Transmit Clock - If set an external 1X clock is used for the transmitter. If cleared the internal baud rate generator provides the transmit clock. The input clock is applied to P1.3 (T×C). The user software is responsible for setting or clearing this flag. External receive clock is enabled by setting PCON.3.

IFS (0A4H) - Interframe Spacing - Determines the number of bit times separating transmitted frames in CSMA/CD. A bit time is equal to 1/baud rate. Only even interframe space periods can be used. The number written into this register is divided by two and loaded in the most significant seven bits. Complete interframe space is obtained by counting this seven bit number down to zero twice. A user software read of this register will give a value where the seven most significant bits gives the current count value and the least significant bit shows a one for the first count-down and a zero for the second count. The value read may not be valid as the timer is clocked in periods not necessarily associated with the CPU read of IFS. Loading this register with zero results in 256 bit times.

MYSLOT (0F5H) - Slot Address Register

7	6	5	4	3	2	1	0
DCJ	DCR	SA5	SA4	SA3	SA2	SA1	SA0
SAn = SLOT ADDRESS (BITS 5 – 0)							

Figure 3.15. MYSLOT

MYSLOT.0, 1, 2, 3, 4, 5 - Slot Address - The six address bits choose 1 of 64 slot addresses. Address 63 has the highest priority and address 1 has the lowest. A value of zero will prevent a station from transmitting during the collision resolution period by waiting until all the possible slot times have elapsed. The user software normally initializes this address in the operating software.

MYSLOT.6 (DCR) - Deterministic Collision Resolution Algorithm - When set, the alternate collision resolution algorithm is selected. Retriggering of the IFS on reappearance of the carrier is also disabled. When using this feature Alternate Backoff Mode must be selected and several other registers must be initialized. User software must initialize TCDCNT with the maximum number of slots that are most appropriate for a particular application. The PRBS register must be set to all ones. This disables the PRBS by freezing it's contents at 0FFH. The backoff timer is used to count down the number of slots based on the slot timer value setting the period of one slot. The user software is responsible for setting or clearing this flag.

MYSLOT.7 (DCJ) - D.C. Jam - When set selects D.C. type jam, when clear, selects A.C. type jam. The user software is responsible for setting or clearing this flag.

PCON (087H)

7	6	5	4	3	2	1	0
SMOD	ARB	REQ	GAREN	XRCLK	GFIEN	PD	IDL

PCON contains bits for power control, LSC control, DMA control, and GSC control. The bits used for the GSC are PCON.2, PCON.3, and PCON.4.

PCON.2 (GFIEN) - GSC Flag Idle Enable - Setting GFIEN to a 1 caused idle flags to be generated between transmitted frames in SDLC mode. SDLC idle flags consist of 01111110 flags creating the sequence 01111110011111110 . . . . . 011111110. A possible side effect of enabling GFIEN is that the maximum possible latency from writing to TFIFO until the first bit is transmitted increased from approximately 2 bit-times to around 8 bit-times. GFIEN has no effect with CSMA/CD.

PCON.3 (XRCLK) - GSC External Receive Clock Enable - Writing a 1 to XRCLK enables an external clock to be applied to pin 5 (Port 1.4). The external clock is used to determine when bits are loaded into the receiver.

PCON.4 (GAREN) - GSC Auxiliary Receiver Enable Bit - This bit needs to be set to a 1 to enable the reception of back-to-back SDLC frames. A back-to-back SDLC frame is when the EOF and BOF is shared between two sequential frames intended for the same station on the link. If GAREN contains a 0 then the receiver will be disabled upon reception of the EOF and by the time user software re-enables the receiver the first bit(s) may have already passed, in the case of back-to-back frames. Setting GAREN to a 1, prevents the receiver from being disabled by the EOF but GREN will be cleared and can be checked by user software to determine that an EOF has been received. GAREN has no effect if the GSC is in CSMA/CD mode.

PRBS (0E4H) - Pseudo-Random Binary Sequence - This register contains a pseudo-random number to be used in the CSMA/CD backoff algorithm. The number is generated by using a feedback shift register clocked by the CPU phase clocks. Writing all ones to the PRBS will freeze the value at all ones. Writing any other value to it will restart the PRBS generator. The PRBS is initialized to all zero's during RESET. A read of location 0E4H will not necessarily give the seed used in the backoff algorithm because the PRBS counters are clocked by internal CPU phase clocks. This means the contents of the PRBS may have been altered between the time when the seed was generated and before a READ has been internally executed.

RFIFO (0F4H) - Receive FIFO - RFIFO is a 3 byte buffer that is loaded each time the GSC receiver has a byte of data. Associated with RFIFO is a pointer that is automatically updated with each read of the FIFO. A read of RFIFO fetches the oldest data in the FIFO.

RSTAT (0E8H) - Receive Status Register

7	6	5	4	3	2	1	0
OR	RCABT	AE	CRCE	RDN	RFNE	GREN	HABEN

Figure 3.16. RSTAT

RSTAT.0 (HABEN) - Hardware Based Acknowledge Enable - If set, enables the hardware based acknowledge feature. The user software is responsible for setting or clearing this flag.

RSTAT.1 (GREN) - Receiver Enable - When set, the receiver is enabled to accept incoming frames. This also clears RDN, CRCE, AE, RCABT, and the receive FIFO. It is cleared by the receiver at the end of a reception or if any errors occurred. The user software is responsible for setting this flag and the GSC or user software can clear it. The status of GREN has no effect on whether the receiver detects a collision in CSMA/CD mode as the receiver input circuitry always monitors the receive pin.

RSTAT.2 (RFNE) - Receive FIFO Not Empty - If set, indicates that the receive FIFO contains data. The receive FIFO is a three byte buffer into which the receive data is loaded. A CPU read of the FIFO retrieves the oldest data and automatically updates the FIFO pointers. Setting GREN to a one will clear the receive FIFO. The status of this flag is controlled by the GSC. It is cleared if user empties receive FIFO.

RSTAT.3 (RDN) - Receive Done - If set, indicates the successful completion of a receiver operation. Will not be set if a CRC, alignment, abort, or FIFO overrun error occurred. The status of this flag is controlled by the GSC.

RSTAT.4 (CRCE) - CRC Error - If set, indicates that a properly aligned frame was received with a mismatched CRC. The status of this flag is controlled by the GSC.

RSTAT.5 (AE) - Alignment Error - If set, indicates that the line went idle when the receiver shift register was not full and the resulting CRC was bad in the CSMA/CD mode. If a correct CRC was valid then AE is not set. In SDLC mode, AE indicates that a non-byte-aligned flag was received. The status of this flag is controlled by the GSC.

RSTAT.6 (RCABT) - Receiver Collision/Abort Detect - If set, indicates that a collision was detected after data had been loaded into the receive FIFO in CSMA/CD mode. In SDLC mode, RCABT indicates that 7 consecutive ones were detected prior to the end flag but after data has been loaded into the receive FIFO. The status of this flag is controlled by the GSC.

RSTAT.7 (OVR) - Overrun - If set, indicates that the receive FIFO was full and new shift register data was written into it. The setting of this flag is controlled by the GSC and it is cleared by user software.

SLOTTM (0BH) - Slot Time - Determines the length of the slot time used in CSMA/CD. A slot time equals $(256 - \text{SLOTTM}) \times (1 / \text{baud rate})$. A read of SLOTTM will give the value of the slot time timer but the value may be invalid as the timer is clocked asynchronously to the CPU. Loading SLOTTM with 0 results in 256 bit times.

TCDCNT (0D4H) - Transmit Collision Detect Count - Contains the number of collisions that have occurred if probabilistic CSMA/CD is used. The user software must clear this register before transmitting a new frame so that the GSC backoff hardware can accurately distinguish a new frame from a retransmit attempt.

In deterministic backoff mode, TCDCNT is used to hold the maximum number of slots.

TFIFO (85H) - GSC Transmit FIFO - TFIFO is a 3 byte buffer with an associated pointer that is automatically updated for each write by user software. Writing a byte to TFIFO loads the data into the next available location in the transmit FIFO. Setting TEN clears the transmit FIFO so the transmit FIFO should not be written to prior to setting TEN. If TEN is already set transmission begins as soon as data is written to TFIFO.

TSTAT (0D8) - Transmit Status Register

7	6	5	4	3	2	1	0
LNI	NOACK	UR	TCDT	TDN	TFNF	TEN	DMA

Figure 3.17. TSTAT

TSTAT.0 (DMA) - DMA Select - If set, indicates that DMA channels are used to service the GSC FIFO's and GSC interrupts occur on TDN and RDN, and also enables UR to become set. If cleared, indicates that the GSC is operating in its normal mode and interrupts occur on TFNF and RFNE. For more information on DMA servicing please refer to the DMA section on DMA serial demand mode (4.2.2.3). The user software is responsible for setting or clearing this flag.

TSTAT.1 (TEN) - Transmit Enable - When set causes TDN, UR, TCDT, and NOACK flag to be reset and the TFIFO cleared. The transmitter will clear TEN after a successful transmission, a collision during the data, CRC, or end flag. The user software is responsible for setting but the GSC or user software may clear this flag. If cleared during a transmission the GSC transmit pin goes to a steady state high level. This is the method used to send an abort character in SDLC. Also $\overline{DEN}$ is forced to a high level. The end of transmission occurs whenever the TFIFO is emptied.

TSTAT.2 (TFNF) - Transmit FIFO not full - When set, indicates that new data may be written into the transmit FIFO. The transmit FIFO is a three byte buffer that loads the transmit shift register with data. The status of this flag is controlled by the GSC.

TSTAT.3 (TDN) - Transmit Done - When set, indicates the successful completion of a frame transmission. If HABEN is set, TDN will not be set until the end of the IFS following the transmitted message, so that the acknowledge can be checked. If an acknowledge is expected and not received, TDN is not set. An acknowledge is not expected following a broadcast or multi-cast packet. The status of this flag is controlled by the GSC.

TSTAT.4 (TCDT) - Transmit Collision Detect - If set, indicates that the transmitter halted due to a collision. It is set if a collision occurs during the data or CRC or if there are more than eight collisions. The status of this flag is controlled by the GSC.

TSTAT.5 (UR) - Underrun - If set, indicates that in DMA mode the last bit was shifted out of the transmit register and that the DMA byte count did not equal zero. When an underrun occurs, the transmitter halts without sending the CRC or the end flag. The status of this flag is controlled by the GSC.

TSTAT.6 (NOACK) - No Acknowledge - If set, indicates that no acknowledge was received for the previous frame. Will be set only if HABEN is set and no acknowledge is received prior to the end of the IFS. NOACK is not set following a broadcast or a multicast packet. The status of this flag is controlled by the GSC.

TSTAT.7 (LNI) - Line Idle - If set, indicates the receive line is idle. In SDLC protocol it is set if 15 consecutive ones are received. In CSMA/CD protocol, line idle is set if no transitions occur on GR×D for approximately 1.6 bit times after a required transition. LNI is cleared after a transition on GR×D. The status of this flag is controlled by the GSC.

3.8 Serial Backplane vs. Network Environment

The C152 GSC port is intended to fulfill the needs of both serial backplane environment and the serial communication network environment. The serial backplane is where typically, only processor to processor communications take place within a self contained box. The communication usually only encompasses those items which are necessary to accomplish the dedicated task for the box. In these types of applications there may not be a need for line drivers as the distance between the transmitter and receiver is relatively short. The network environment; however, usually requires transmission of data over large distances and requires drivers and/or repeaters to ensure the data is received on both ends.

4.0 DMA Operation

The C152 contains DMA (Direct Memory Accessing) logic to perform high speed data transfers between any two of

Internal Data RAM
Internal SFRs
External Data RAM

If external RAM is involved, the Port 2 and Port 0 pins are used as the address/data bus, and $\overline{RD}$ and $\overline{WR}$ signals are generated as required.

Hardware is also implemented to generate a Hold Request signal and await a Hold Acknowledge response before commencing a DMA that involves external RAM.

Alternatively, the Hold/Hold Acknowledge hardware can be programmed to accept a Hold Request signal from an external device and generate a Hold Acknowledge signal in response, to indicate to the requesting device that the C152 will not commence a DMA to or from external RAM while the Hold Request is active.

4.1 DMA with the 80C152

The C152 contains two identical general purpose 8-bit DMA channels with 16-bit addressability: DMA0 and DMA1. DMA transfers can be executed by either channel independent of the other, but only by one channel at a time. During the time that a DMA transfer is being executed, program execution is suspended. A DMA transfer takes one machine cycle (12 oscillator

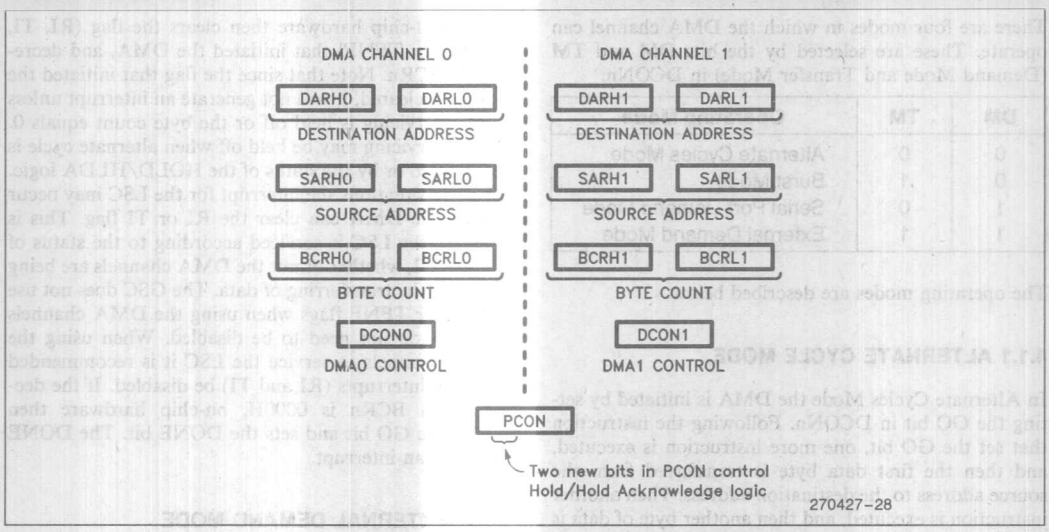


Figure 4.1. DMA Registers

periods) per byte transferred, except when the destination and source are both in External Data RAM. In that case the transfer takes two machine cycles per byte. The term DMA Cycle will be used to mean the transfer of a single data byte, whether it takes 1 or 2 machine cycles.

Associated with each channel are seven SFRs, shown in Figure 4.1. SARLn and SARHn holds the low and high bytes of the source address. Taken together they form a 16-bit Source Address Register. DARLn and DARHn hold the low and high bytes of the destination address, and together form the Destination Address Register. BCRLn and BCRHn hold the low and high bytes of the number of bytes to be transferred, and together form the Byte Count Register. DCONn contains control and flag bits.

Two bits in DCONn are used to specify the physical destination of the data transfer. These bits are DAS (Destination Address Space) and IDA (Increment Destination Address). If DAS = 0, the destination is in data memory external to the C152. If DAS = 1, the destination is internal to the C152. If DAS = 1 and IDA = 0, the internal destination is a Special Function Register (SFR). If DAS = 1 and IDA = 1, the internal destination is in the 256-byte data RAM.

In any case, if IDA = 1, the destination address is automatically incremented after each byte transfer. If IDA = 0, it is not.

Two other bits in DCONn specify the physical source of the data to be transferred. These are SAS (Source Address Space) and ISA (Increment Source Address). If SAS = 0, the source is in data memory external to the C152. If SAS = 1, the source is internal. If SAS = 1 and ISA = 0, the internal source is an SFR. If SAS = 1 and ISA = 1, the internal source is in the 256-byte data RAM.

In any case, if ISA = 1, the source address is automatically incremented after each byte transfer. If ISA = 0, it is not.

The functions of these four control bits are summarized below:

DAS	IDA	Destination	Auto-Increment
0	0	External RAM	no
0	1	External RAM	yes
1	0	SFR	no
1	1	Internal RAM	yes
SAS	**ISA**	**Source**	**Auto-Increment**
0	0	External RAM	no
0	1	External RAM	yes
1	0	SFR	no
1	1	Internal RAM	yes

There are four modes in which the DMA channel can operate. These are selected by the bits DM and TM (Demand Mode and Transfer Mode) in DCONn:

DM	TM	Operating Mode
0	0	Alternate Cycles Mode
0	1	Burst Mode
1	0	Serial Port Demand Mode
1	1	External Demand Mode

The operating modes are described below.

4.1.1 ALTERNATE CYCLE MODE

In Alternate Cycles Mode the DMA is initiated by setting the GO bit in DCONn. Following the instruction that set the GO bit, one more instruction is executed, and then the first data byte is transferred from the source address to the destination address. Then another instruction is executed, and then another byte of data is transferred, and so on in this manner.

Each time a data byte is transferred, BCRn (Byte Count Register for DMA Channel n) is decremented. When it reaches 0000H, on-chip hardware clears the GO bit and sets the DONE bit, and the DMA ceases. The DONE bit flags an interrupt.

4.1.2 BURST MODE

Burst Mode differs from Alternate Cycles mode only in that once the data transfer has begun, program execution is entirely suspended until BCRn reaches 0000H, indicating that all data bytes that were to be transferred have been transferred. The interrupt control hardware remains active during the DMA, so interrupt flags may get set, but since program execution is suspended, the interrupts will not be serviced while the DMA is in progress.

4.1.3 SERIAL PORT DEMAND MODE

In this mode the DMA can be used to service the Local Serial Channel (LSC) or the Global Serial Channel (GSC).

In Serial Port Demand Mode the DMA is initiated by any of the following conditions, if the GO bit is set:

Source Address = SBUF	.AND.	RI = 1
Destination Address = SBUF	.AND.	TI = 1
Source Address = RFIFO	.AND.	RFNE = 1
Destination Address = TFIFO	.AND.	TFNF = 1

Each time one of the above conditions is met, one DMA Cycle is executed; that is, one data byte is transferred from the source address to the destination address. On-chip hardware then clears the flag (RI, TI, RFNE, or TFNF) that initiated the DMA, and decrements BCRn. Note that since the flag that initiated the DMA is cleared, it will not generate an interrupt unless DMA servicing is held off or the byte count equals 0. DMA servicing may be held off when alternate cycle is being used or by the status of the HOLD/HLDA logic. In these situations the interrupt for the LSC may occur before the DMA can clear the RI or TI flag. This is because the LSC is serviced according to the status of RI and TI, whether or not the DMA channels are being used for the transferring of data. The GSC does not use RFNE or TFNF flags when using the DMA channels so these do not need to be disabled. When using the DMA channels to service the LSC it is recommended that the interrupts (RI and TI) be disabled. If the decremented BCRn is 0000H, on-chip hardware then clears the GO bit and sets the DONE bit. The DONE bit flags an interrupt.

4.1.4 EXTERNAL DEMAND MODE

In External Demand Mode the DMA is initiated by one of the External Interrupt pins, provided the GO bit is set. $\overline{INT0}$ initiates a Channel 0 DMA, and $\overline{INT1}$ initiates a Channel 1 DMA.

If the external interrupt is configured to be transition-activated, then each 1-to-0 transition at the interrupt pin sets the corresponding external interrupt flag, and generates one DMA Cycle. Then, BCRn is decremented. No more DMA Cycles take place until another 1-to-0 transition is seen at the external interrupt pin. If the decremented BCRn = 0000H, on-chip hardware clears the GO bit and sets the DONE bit. If the external interrupt is enabled, it will be serviced.

If the external interrupt is configured to be level-activated, then DMA Cycles commence when the interrupt pin is pulled low, and continue for as long as the pin is held low and BCRn is not 0000H. If BCRn reaches 0 while the interrupt pin is still low, the GO bit is cleared, the DONE bit is set, and the DMA ceases. If the external interrupt is enabled, it will be serviced.

If the interrupt pin is pulled up before BCRn reaches 0000H, then the DMA ceases, but the GO bit is still 1 and the DONE bit is still 0. An external interrupt is not generated in this case, since in level-activated mode, pulling the pin to a logical 1 clears the interrupt flag. If the interrupt pin is then pulled low again, DMA transfers will continue from where they were previously stopped.

The timing for the DMA Cycle in the transition-activated mode, or for the first DMA Cycle in the level-activated mode is as follows: If the 1-to-0 transition is

detected before the final machine cycle of the instruction in progress, then the DMA commences as soon as the instruction in progress is completed. Otherwise, one more instruction will be executed before the DMA starts. No instruction is executed during any DMA Cycle.

4.2 Timing Diagrams

Timing diagrams for single-byte DMA transfers are shown in Figures 4.2 through 4.5 for four kinds of DMA Cycles: internal memory to internal memory, internal memory to external memory, external memory to internal memory, and external memory to external memory. In each case we assume the C152 is executing out of external program memory. If the C152 is executing out of internal program memory, then $\overline{PSEN}$ is inactive, and the Port 0 and Port 2 pins emit P0 and P2 SFR data. If External Data Memory is involved, the Port 0 and Port 2 pins are used as the address/data bus, and $\overline{RD}$ and/or $\overline{WR}$ signals are generated as needed, in the same manner as in the execution of a MOVX @DPTR instruction.

4.3 Hold/Hold Acknowledge

Two operating modes of Hold/Hold Acknowledge logic are available, and either or neither may be invoked by software. In one mode, the C152 generates a Hold Request signal and awaits a Hold Acknowledge response before commencing a DMA that involves external RAM. This is called the Requester Mode.

In the other mode, the C152 accepts a Hold Request signal from an external device and generates a Hold Acknowledge signal in response, to indicate to the requesting device that the C152 will not commence a DMA to or from external RAM while the Hold Request is active. This is called the Arbiter mode.

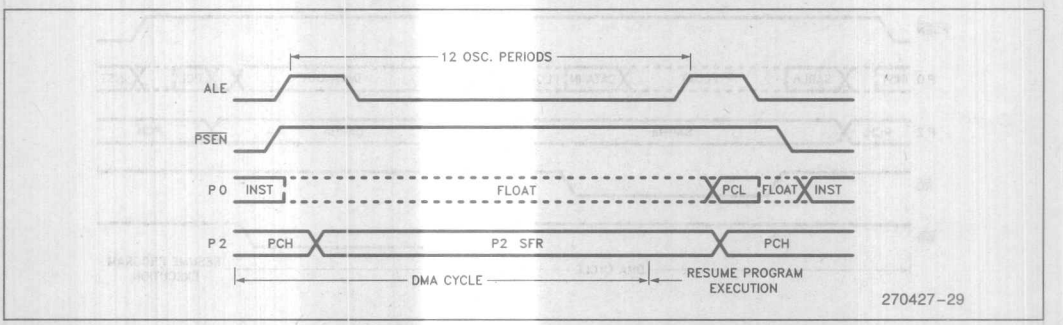


Figure 4.2. DMA Transfer from Internal Memory to Internal Memory

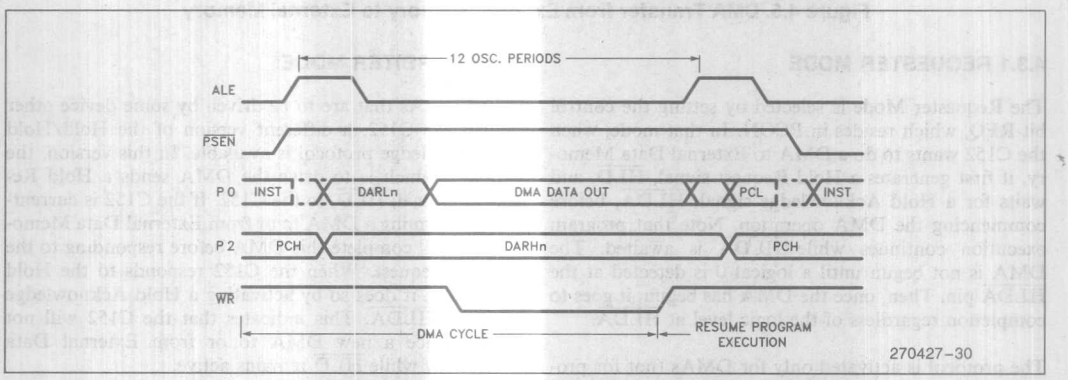


Figure 4.3. DMA Transfer from Internal Memory to External Memory

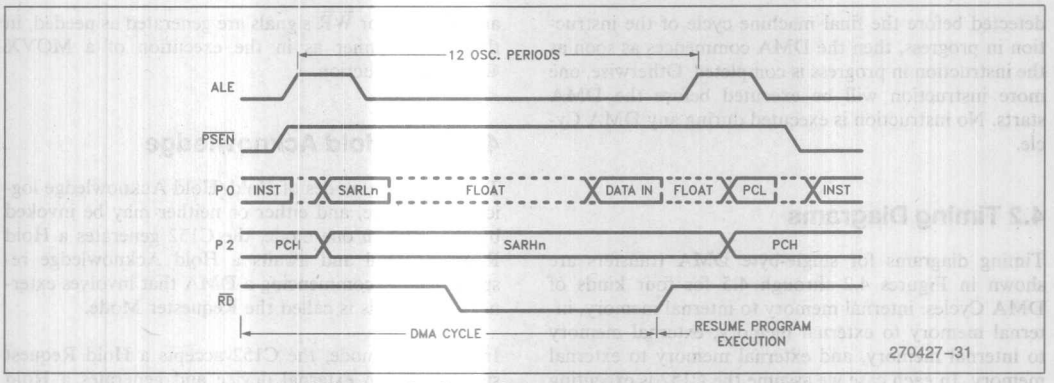


Figure 4.4. DMA Transfer from External Memory to Internal Memory

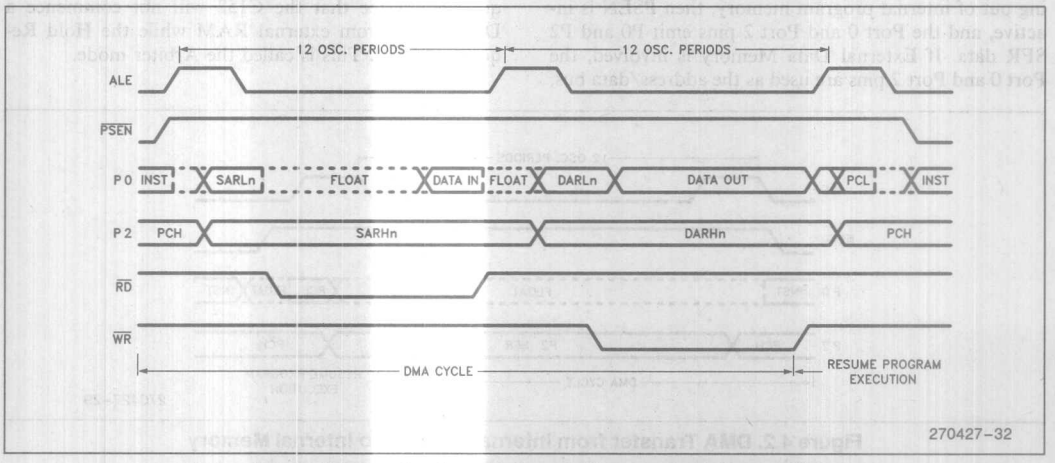


Figure 4.5. DMA Transfer from External Memory to External Memory

4.3.1 REQUESTER MODE

The Requester Mode is selected by setting the control bit REQ, which resides in PCON. In that mode, when the C152 wants to do a DMA to External Data Memory, it first generates a Hold Request signal, $\overline{\text{HLD}}$, and waits for a Hold Acknowledge signal, $\overline{\text{HLDA}}$, before commencing the DMA operation. Note that program execution continues while $\overline{\text{HLDA}}$ is awaited. The DMA is not begun until a logical 0 is detected at the $\overline{\text{HLDA}}$ pin. Then, once the DMA has begun, it goes to completion regardless of the logic level at $\overline{\text{HLDA}}$.

The protocol is activated only for DMAs (not for program fetches or MOVX operations), and only for DMAs to or from External Data Memory. If the data destination and source are both internal to the C152, the $\overline{\text{HLD}}$/$\overline{\text{HLDA}}$ protocol is not used.

The $\overline{\text{HLD}}$ output is an alternate function of port pin P1.5, and the $\overline{\text{HLDA}}$ input is an alternate function of port pin P1.6.

4.3.2 ARBITER MODE

For DMAs that are to be driven by some device other than the C152, a different version of the Hold/Hold Acknowledge protocol is available. In this version, the device which is to drive the DMA sends a Hold Request signal, $\overline{\text{HLD}}$, to the C152. If the C152 is currently performing a DMA to or from External Data Memory, it will complete this DMA before responding to the Hold Request. When the C152 responds to the Hold Request, it does so by activating a Hold Acknowledge signal, $\overline{\text{HLDA}}$. This indicates that the C152 will not commence a new DMA to or from External Data Memory while $\overline{\text{HLD}}$ remains active.

Note that in the Arbiter Mode the C152 does not suspend program execution at all, even if it is executing from external program memory. It does not surrender use of its own bus.

The Hold Request input, $\overline{\text{HLD}}$, is at P1.5. The Hold Acknowledge output, $\overline{\text{HLDA}}$, is at P1.6. This

version of the Hold/Hold Acknowledge feature is selected by setting the control bit ARB in PCON.

The functions of the ARB and REQ bits in PCON, then, are

ARB	REQ	Hold/Hold Acknowledge Logic
0	0	Disabled
0	1	C152 generates $\overline{HLD}$, detects $\overline{HLDA}$
1	0	C152 detects $\overline{HLD}$, generates $\overline{HLDA}$
1	1	Invalid

4.3.3 USING THE HOLD/HOLD ACKNOWLEDGE

The $\overline{HOLD}/\overline{HOLDA}$ logic only affects DMA operation with external RAM and doesn't affect other operations with external RAM, such as MOVX instruction.

Figure 4.6 shows a system in which two 83C152s are sharing a global RAM. In this system, both CPUs are executing from internal ROM. Neither CPU uses the bus except to access the shared RAM, and such accesses are done only through DMA operations, not by MOVX instructions.

One CPU is programmed to be the Arbiter and the other, to be the Requester. The ALE Switch selects which CPU's ALE signal will be directed to the address latch. The Arbiter's ALE is selected if $\overline{HLDA}$ is high, and the Requester's ALE is selected if $\overline{HLDA}$ is low.

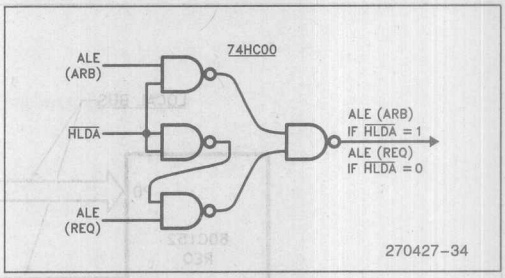


Figure 4.7. ALE Switch Select

The ALE Switch logic can be implemented by a single 74HC00, as shown in Figure 4.7.

Figure 4.8 shows the additional gating that is required when one of the CPUs is executing from external ROM. In this case the CPU has constant access to its own local bus, and accesses the global RAM only after gaining access to a global address/data bus.

If the CPU is programmed to be a Requester, as shown in Figure 4.8, then when it wants to access the global RAM it first activates $\overline{HLD}$, and continues program execution on its own local bus while awaiting an active level at $\overline{HLDA}$.

An active level at $\overline{HLDA}$ enables the local bus to the global bus, and signals the Requester to proceed with its DMA. The Requester's $\overline{RD}$ signal, rather than its

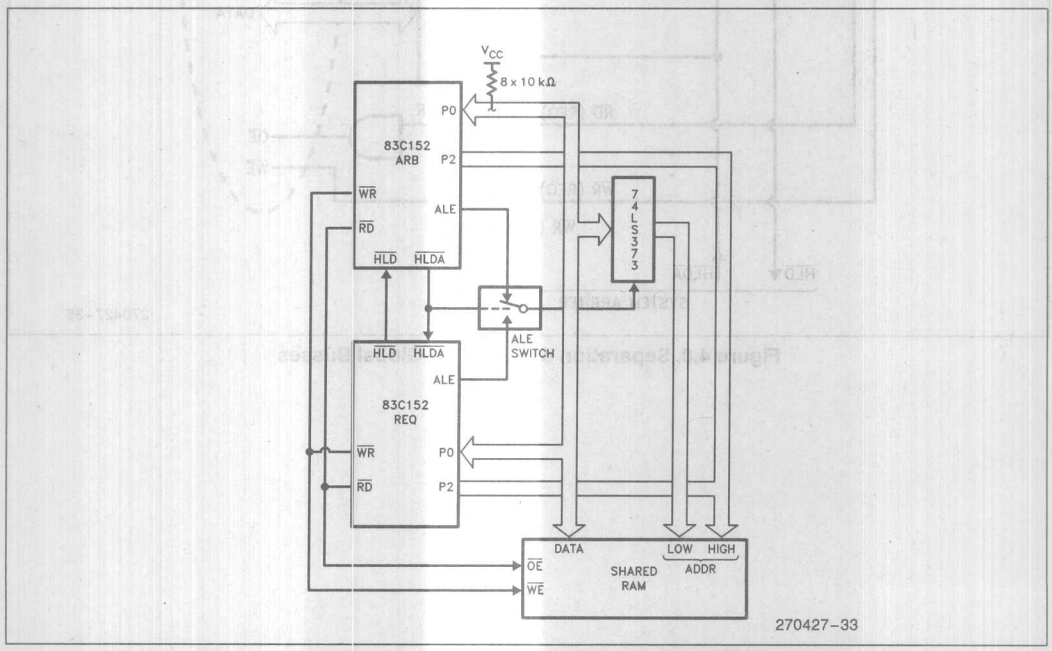


Figure 4.6. Two 83C152s Sharing External RAM

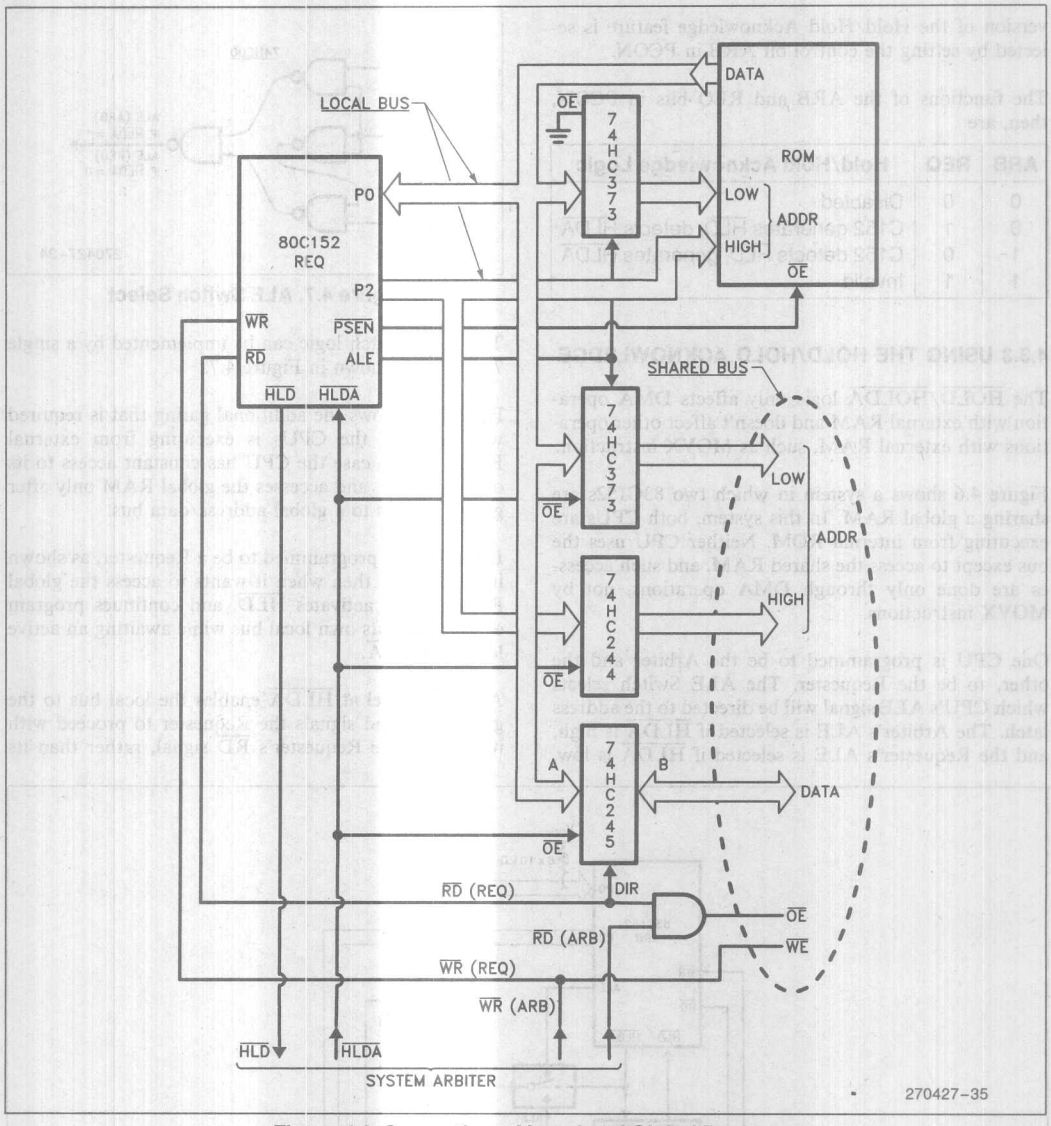


Figure 4.8. Separation of Local and Global Busses

$\overline{WR}$ signal, is used to control the direction of the transceiver. This is to ensure that the transceiver will not try to drive the local bus before the DMA has actually begun.

$\overline{RD}$ from the Requester is logically ANDed with $\overline{RD}$ from the Arbiter to activate $\overline{OE}$ to the RAM. $\overline{WR}$ from the Requester can normally be hard-wired to $\overline{WR}$ from the Arbiter to activate $\overline{WE}$ to the RAM.

4.4 DMA Arbitration

The DMA Arbitration described in this section is not arbitration between two devices wanting to access a shared RAM, but rather on-chip arbitration between the two DMA channels on the C152.

The C152 provides two DMA channels, either of which may be called into operation at any time in response to real time conditions in the application circuit. However, only one DMA channel can be serviced during a single DMA cycle.

In the event that both DMA channels request service at the same time, DMA Channel 0 takes precedence.

Figure 4.9 shows the three tasks to which the internal bus of the C152 can be dedicated. In this figure, Instruction Cycle means the complete execution of a single instruction, whether it takes 1, 2 or 4 machine cycles. DMA Cycle means the transfer of a single data byte from source to destination, whether it takes 1 or 2 machine cycles. (It takes 2 machine cycles if the destination and source are both external to the C152.) On-chip arbitration logic determines which type of cycle is executed, according to the following rules.

If the $\overline{HLD}/\overline{HLDA}$ logic is disabled (ARB = 0, REQ = 0):

* A write to any DMA address or control register is always followed by an Instruction Cycle. If the next instruction is a read or write to a DMA address or control register, the DMA cycle is held off one more instruction cycle.
* A DMA0 Cycle is called for if GO0 = 1 and any of the following conditions are satisfied:
  1. Channel 0 Burst Mode is selected;
  2. Channel 0 is in SP Demand Mode and a SP Demand flag is up (but see **);

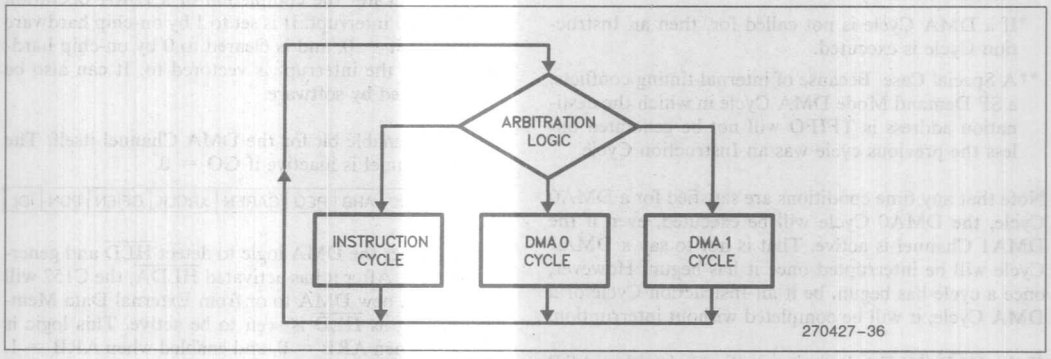


Figure 4.9. Internal Bus Tasks

External Demand flag is up;

4. Channel 0 is in Alternate Cycles Mode and Channel 1 isn't, and the previous cycle was not a DMA Cycle;
5. Channel 0 and Channel 1 are both in Alternate Cycles Mode, and the previous cycle was not a DMA Cycle, and the previous DMA Cycle was not a DMA0 cycle.

* A DMA1 Cycle is called for if GO1 = 1 and no condition for a DMA0 Cycle is satisfied, and any of the following conditions are satisfied:

1. Channel 1 Burst Mode is selected;
2. Channel 1 is in SP Demand Mode and a SP Demand flag is up (but see **);
3. Channel 1 is in External Demand Mode and an External Demand flag is up;
4. Channel 1 is in Alternate Cycles Mode and Channel 0 isn't, and the previous cycle was not a DMA Cycle;
5. Channel 1 and Channel 0 are both in Alternate Cycles Mode, and the previous cycle was not a DMA Cycle, and the previous DMA Cycle was not a DMA1 cycle.

*If a DMA Cycle is not called for, then an Instruction Cycle is executed.

**A Special Case: Because of internal timing conflicts, a SP Demand Mode DMA Cycle in which the destination address is TFIFO will not be generated unless the previous cycle was an Instruction Cycle.

Note that any time conditions are satisfied for a DMA0 Cycle, the DMA0 Cycle will be executed, even if the DMA1 Channel is active. That is not to say a DMA1 Cycle will be interrupted once it has begun. However, once a cycle has begun, be it an Instruction Cycle or a DMA Cycle, it will be completed without interruption.

If the $\overline{HLD}$/$\overline{HLDA}$ logic is not disabled (either ARB = 1 or REQ = 1), then the Hold/Hold Acknowledge protocol will also be observed, as previously described, for DMAs to or from external RAM.

4.5 Summary of DMA Control Bits

DCONn	DAS	IDA	SAS	ISA	DM	TM	DONE	GO

DAS specifies the Destination Address Space. If DAS = 0, the destination is in External Data Memory. If DAS = 1 and IDA = 0, the destination is a Special Function Register (SFR). If DAS = 1 and IDA = 1, the destination is in Internal Data RAM.

destination address is automatically incremented after each byte transfer. If IDA = 0, it is not.

SAS specifies the Source Address Space. If SAS = 0, the source is in External Data Memory. If SAS = 1 and ISA = 0, the source is an SFR. If SAS = 1 and ISA = 1, the source is Internal Data RAM.

ISA (Increment Source Address) If ISA = 1, the source address is automatically incremented after each byte transfer. If ISA = 0, it is not.

DM (Demand Mode) If DM = 1, the DMA Channel operates in Demand Mode. In Demand Mode the DMA is initiated either by an external signal or by a Serial Port flag, depending on the value of the TM bit. If DM = 0, the DMA is requested by setting the GO bit in software.

TM (Transfer Mode) If DM = 1 then TM selects whether a DMA is initiated by an external signal (TM = 1) or by a Serial Port flag (TM = 0). If DM = 0 then TM selects whether the data transfers are to be in bursts (TM = 1) or in alternate cycles (TM = 0).

DONE indicates the completion of a DMA operation and flags an interrupt. It is set to 1 by on-chip hardware when BCRn = 0, and is cleared to 0 by on-chip hardware when the interrupt is vectored to. It can also be set or cleared by software.

GO is the enable bit for the DMA Channel itself. The DMA Channel is inactive if GO = 0.

PCON	SMOD	ARB	REQ	GAREN	XRCLK	GFIEN	PDN	IDL

ARB enables the DMA logic to detect $\overline{HLD}$ and generate $\overline{HLDA}$. After it has activated $\overline{HLDA}$, the C152 will not begin a new DMA to or from External Data Memory as long as $\overline{HLD}$ is seen to be active. This logic is disabled when ARB = 0, and enabled when ARB = 1.

REQ enables the DMA logic to generate $\overline{HLD}$ and detect $\overline{HLDA}$ before performing a DMA to or from External Data Memory. After it has activated $\overline{HLD}$, the C152 will not begin the DMA until $\overline{HLDA}$ is seen to be active. This logic is disabled when REQ = 0, and enabled when REQ = 1.

5.0 GLOSSARY

ADR0,1,2,3 (95H, 0A5H, 0B5H, 0C5H) - Address Match Registers 0,1,2,3 - The contents of these SFRs are compared against the address bits from the serial

data on the GSC. If the address matches the SFR, then the C152 accepts that frame. If in 8 bit addressing mode, a match with any of the four registers will trigger acceptance. In 16 bit addressing mode, a match with ADR1:ADR0 or ADR3:ADR2 will be accepted. Address length is determined by GMOD (AL).

AE - Alignment Error, see RSTAT.

AL - Address Length, see GMOD.

AMSK0,1 (0D5H, 0E5H) - Address Match Mask 0,1 - Identifies which bits in ADR0,1 are "don't care" bits. Setting a bit to 1 in AMSK0,1 identifies the corresponding bit in ADDR0,1 as not to be examined when comparing addresses.

BAUD - (94H) Contains the programmable value for the baud rate generator for the GSC. The baud rate will equal (fosc)/((BAUD + 1) × 8).

BCRL0,1 (0E2H, 0F2H) - Byte Count Register Low 0,1 - Contains the lower byte of the byte count. Used during DMA transfers to identify to the DMA channels when the transfer is complete.

BCRH0,1 (0E3H, 0F3H) - Byte Count Register High 0,1 - Contains the upper byte of the byte count.

BKOFF (0C4H) - Backoff Timer - The backoff timer is an eight bit count-down timer with a clock period equal to one slot time. The backoff time is used in the CSMA/CD collision resolution algorithm.

BOF - Beginning of Frame flag - A term commonly used when dealing with packetized data. Signifies the beginning of a frame.

CRC - Cyclic Redundancy Check - An error checking routine that mathematically manipulates a value dependent on the incoming data. The purpose is to identify when a frame has been received in error.

CRCE - CRC Error, see RSTAT.

CSMA/CD - Stands for Carrier Sense, Multiple Access, with Collision Detection.

CT - CRC Type, see GMOD.

DARL0/1 (0C2H, 0D2H) - Destination Address Register Low 0/1 - Contains the lower byte of the destinations' address when performing DMA transfers.

DARH0/1 (0C3H, 0D3H) - Destination Address Register Low 0/1 - Contains the upper byte of the destinations' address when performing DMA transfers.

DAS - Destination Address Space, see DCON.

DCJ - D.C. Jam, see MYSLOT.

DCON0/1 (092H,093H)

7	6	5	4	3	2	1	0
DAS	IDA	SAS	ISA	DM	TM	DONE	GO

The DCON registers control the operation of the DMA channels by determining the source of data to be transferred, the destination of the data to be transfer, and the various modes of operation.

DCON.0 (GO) - Enables DMA Transfer - When set it enables a DMA channel. If block mode is set then DMA transfer starts as soon as possible under CPU control. If demand mode is set then DMA transfer starts when a demand is asserted and recognized.

DCON.1 (DONE) - DMA Transfer is Complete - When set the DMA transfer is complete. It is set when BCR equals 0 and is automatically reset when the DMA vectors to its interrupt routine. If DMA interrupt is disabled and the user software executes a jump on the DONE bit, then the user software must also reset the done bit. If DONE is not set, then the DMA transfer is not complete.

DCON.2 (TM) - Transfer Mode - When set, DMA burst transfers are used if the DMA channel is configured in block mode or external interrupts are used to initiate a transfer if in Demand Mode. When TM is cleared, Alternate Cycle Transfers are used if DMA is in the Block Mode, or Local Serial channel/GSC interrupts are used to initiate a transfer if in Demand Mode.

DCON.3 (DM) - DMA Channel Mode - When set, Demand Mode is used and when cleared, Block Mode is used.

DCON.4 (ISA) - Increment Source Address - When set, the source address registers are automatically incremented during each transfer. When cleared, the source address registers are not incremented.

DCON.5 (SAS) - Source Address Space - When set, the source of data for the DMA transfers is internal data memory if autoincrement is also set. If autoincrement is not set but SAS is, then the source for data will be one of the Special Function Registers. When SAS is cleared, the source for data is external data memory.

DCON.6 (IDA) - Increment Destination Address Space - When set, destination address registers are incremented once after each byte is transferred. When cleared, the destination address registers are not automatically incremented.

DCON.7 (DAS) - Destination Address Space - When set, destination of data to be transferred is internal data memory if autoincrement mode is also set. If autoincrement is not set the destinationwill be one of the Special Function Registers. When DAS is cleared then the destination is external data memory.

DCR - Deterministic Resolution, see MYSLOT.

$\overline{\text{DEN}}$ - An alternate function of one of the port 1 pins (P1.2). Its purpose is to enable external drivers when the GSC is transmitting data. This function is always active when using the GSC and if P1.2 is programmed to a 1.

DM - DMA Mode, see DCON0.

DMA - Direct Memory Access mode, see TSTAT.

DONE - DMA done bit, see DCON0.

DPH - Data Pointer High, an SFR that contains the high order byte of a general purpose pointer called the data pointer (DPTR).

DPL - Data Pointer Low, an SFR that contains the low order byte of the data pointer.

EDMA0 - Enable DMA Channel 0 interrupt, see IEN1.

EDMA1 - Enable DMA Channel 1 interrupt, see IEN1.

EGSRE - Enable GSC Receive Error interrupt, see IEN1.

EGSRV - Enable GSC Receive Valid interrupt, see IEN1.

EGSTE - Enable GSC Transmit Error interrupt, see IEN1.

EGSTV - Enable GSC Transmit Valid interrupt, see IEN1.

EOF - A general term used in serial communications, EOF stands for End Of Frame and signifies when the last bits of data are transmitted when using packetized data.

ES - Enable LSC Service interrupt, see IE.

ET0 - Enable Timer 0 interrupt, see IE.

ET1 - Enable Timer 1 interrupt, see IE.

EX0 - Enable External interrupt 0, see IE.

EX1 - Enable External interrupt 1, see IE.

GMOD (84H)

7	6	5	4	3	2	1	0
XTCLK	M1	M0	AL	CT	PL1	PL0	PR

The bits in this SFR, perform most of the configuration on the type of data transfers to be used with the GSC. Determines the mode, address length, preamble length, protocol select, and enables the external clocking of the transmit data.

GMOD.0 (PR) - Protocol - If set, SDLC protocols with NRZI encoding, zero bit insertion, and SDLC flags are used. If cleared, CSMA/CD link access with Manchester encoding is used.

GMOD.1,2 (PL0,1) - Preamble length

PL1	PL0	LENGTH (BITS)
0	0	0
0	1	8
1	0	32
1	1	64

The length includes the two bit Begin Of frame (BOF) flag in CSMA/CD but does not include the SDLC flag. In SDLC mode, the BOF is an SDLC flag, otherwise it is two consecutive ones. Zero length is not compatible in CSMA/CD mode.

GMOD.3 (CT) - CRC Type - If set, 32-bit AUTODIN-II-32 is used. If cleared, 16-bit CRC-CCITT is used.

GMOD.4 (AL) - Address Length - If set, 16-bit addressing is used. If cleared, 8-bit addressing is used. In 8-bit mode, a match with any of the 4 address registers will allow that frame to be accepted (ADR0, ADR1, ADR2, ADR3). "Don't Care" bits may be masked in ADR0 and ADR1 with AMSK0 and AMSK1. In 16-bit mode, addresses are matched against "ADR1:ADR0" or "ADR3:ADR2". Again, "Don't Care" bits in ADR1:ADR0 can be masked in AMSK1:AMSK0. A received address of all ones will always be recognized in any mode.

GMOD.5, 6 (M0,M1) - Mode Select - Two test modes, an optional "alternate backoff" mode, or normal backoff can be enabled with these two bits.

M1	M0	Mode
0	0	Normal
0	1	Raw Transmit
1	0	Raw Receive
1	1	Alternate Backoff

GMOD.7 (XTCLK) - External Transmit Clock - If set an external 1X clock is used for the transmitter. If cleared the internal baud rate generator provides the

transmit clock. The input clock is applied to P1.3 ($\overline{TxC}$). The user software is responsible for setting or clearing this flag. External receive clock is enabled by setting PCON.3.

GO - DMA Go bit, see DCON0.

GRxD - GSC Receive Data input, an alternate function of one of the port 1 pins (P1.0). This pin is used as the receive input for the GSC. P1.0 must be programmed to a 1 for this function to operate.

GSC - Global Serial Channel - A high-level, multi-protocol, serial communication controller added to the 80C51BH core to accomplish high-speed transfers of packetized serial data.

GTxD - GSC Transmit Data output, an alternate function of one of the port 1 pins (P1.1). This pin is used as the transmit output for the GSC. P1.1 must be programmed to a 1 for this function to operate.

HBAEN - Hardware Based Acknowledge Enable, see RSTAT.

HLDA - Hold Acknowledge, an alternate function of one of the port 1 pins (P1.6). This pin is used to perform the "HOLD ACKNOWLEDGE" function for DMA transfers. HLDA can be an input or an output, depending on the configuration of the DMA channels. P1.6 must be programmed to a 1 for this function to operate.

HOLD - Hold, an alternate function of one of the port 1 pins (P1.5). This pin is used to perform the "HOLD" function for DMA transfers. HOLD can be an input or an output, depending on the configuration of the DMA channels. P1.5 must be programmed to a 1 for this function to operate.

IDA - Increment Destination Address, see DCON0.

IE (0A8H)

7	6	5	4	3	2	1	0
EA			ES	ET1	EX1	ET0	EX0

Interrupt Enable SFR, used to individually enable the Timer and Local Serial Channel interrupts. Also contains the global enable bit which must be set to a 1 to enable any interrupt to be automatically recognized by the CPU.

IE.0 (EX0) - Enables the external interrupt $\overline{INT0}$ on P3.2.

IE.1 (ET0) - Enables the Timer 0 interrupt.

IE.2 (EX1) - Enables the external interrupt $\overline{INT1}$ on P3.3.

IE.3 (ET1) - Enables the Timer 1 interrupt.

IE.4 (ES) - Enables the Local Serial Channel interrupt.

IE.7 (EA) - The global interrupt enable bit. This bit must be set to a 1 for any other interrupt to be enabled.

IEN1 - (0C8H)

7	6	5	4	3	2	1	0
		EGSTE	EDMA1	EGSTV	EDMA0	EGSRE	EGSRV

Interrupt enable register for DMA and GSC interrupts. A 1 in any bit position enables that interrupt.

IEN1.0 (EGSRV) - Enables the GSC valid receive interrupt.

IEN1.1 (EGSRE) - Enables the GSC receive error interrupt.

IEN1.2 (EDMA0) - Enables the DMA done interrupt for Channel 0.

IEN1.3 (EGSTV) - Enables the GSC valid transmit interrupt.

IEN1.4 (EDMA1) - Enables the DMA done interrupt for Channel 1.

IEN1.5 (EGSTE) - Enables the GSC transmit error interrupt

IFS - (0A4H) Interframe Space, determines the number of bit times separating transmitted frames.

IP (0B8H)

7	6	5	4	3	2	1	0
			PS	PT1	PX1	PT0	PX0

Allows the user software two levels of prioritization to be assigned to each of the interrupts in IE. A 1 assigns the corresponding interrupt in IE a higher interrupt than an interrupt with a corresponding 0.

IP.0 (PX0) - Assigns the priority of external interrupt, $\overline{INT0}$.

IP.1 (PT0) - Assigns the priority of Timer 0 interrupt, T0.

IP.2 (PX1) - Assigns the priority of external interrupt, $\overline{INT1}$.

IP.3 (PT1) - Assigns the priority of Timer 1 interrupt, T1.

IP.4 (PS) - Assigns the priority of the LSC interrupt, SBUF.

IPN1 - (0F8H)

7	6	5	4	3	2	1	0
		PGSTE	PDMA1	PGSTV	PDMA0	PGSRE	PGSRV

Allows the user software two levels of prioritization to be assigned to each of the interrupts in IEN1. A 1 assigns the corresponding interrupt in IEN1 a higher interrupt than an interrupt with a corresponding 0.

IPN1.0 (PGSRV) - Assigns the priority of GSC receive valid interrupt.

IPN1.1 (PGSRE) - Assigns the priority of GSC error receive interrupt.

IPN1.2 (PDMA0) - Assigns the priority of DMA done interrupt for Channel 0.

IPN1.3 (PGSTV) - Assigns the priority of GSC transmit valid interrupt.

IPN1.4 (PDMA1) - Assigns the priority of DMA done interrupt for Channel 1.

IPN1.5 (PGSTE) - Assigns the priority of GSC transmit error interrupt.

ISA - Increment Source Address, see DCON0.

LNI - Line Idle, see TSTAT.

LSC - Local Serial Channel - The asynchronous serial port found on all MCS-51 devices. Uses start/stop bits and can transfer only 1 byte at a time.

M0 - One of two GSC mode bits, see TMOD.

M1 - One of two GSC mode bits, see TMOD

MYSLOT - (0F5H)

7	6	5	4	3	2	1	0
DCJ	DCR	SA5	SA4	SA3	SA2	SA1	SA0

Determines which type of Jam is used, which backoff algorithm is used, and the DCR slot address for the GSC.

MYSLOT.0,1,2,3,4,5 (SA0,1,2,3,4,5) - These bits determine which slot address is assigned to the C152 when using deterministic backoff during CSMA/CD operations on the GSC. Maximum slots available is 63. An address of 00H prevents that station from participating in the backoff process.

MYSLOT.6 (DCR) - Determines which collision resolution algorithm is used. If set to a 1, then the deterministic backoff is used. If cleared, then a random slot assignment is used.

MYSLOT.7 (DCJ) - Determines the type of Jam used during CSMA/CD operation when a collision occurs. If set to a 1 then a low D.C. level is used as the jam signal. If cleared, then $\overline{CRC}$ is used as the jam signal. The jam is applied for a length of time equal to the CRC length.

NOACK - No Acknowledgment error bit, see TSTAT.

NRZI - Non-Return to Zero inverted, a type of data encoding where a 0 is represented by a change in the level of the serial link. A 1 is represented by no change.

OVR - Overrun error bit, see RSTAT.

PR - Protocol select bit, see GMOD. PCON (87H)

7	6	5	4	3	2	1	0
SMOD	ARB	REQ	GAREN	XRCLK	GFIEN	PD	IDL

PCON.0 (IDL) - Idle bit, used to place the C152 into the idle power saving mode.

PCON.1 (PD) - Power Down bit, used to place the C152 into the power down power saving mode.

PCON.2 (GFIEN) - GSC Flag Idle Enable bit, when set, enables idle flags (01111110) to be generated between transmitted frames in SDLC mode.

PCON.3 (XRCLK) - External Receive Clock bit, used to enable an external clock to be used for only the receiver portion of the GSC.

PCON.4 (GAREN) - GSC Auxiliary Receive Enable bit, used to enable the GSC to receive back-to-back SDLC frames. This bit has no effect in CSMA/CD mode.

PCON.5 (REQ) - Requester mode bit, set to a 1 when C152 is to be operated as the requester station during DMA transfers.

PCON.6 (ARB) - Arbiter mode bit, set to a 1 when C152 is to be operated as the arbiter during DMA transfers.

PCON.7 (SMOD) - LSC mode bit, used to double the baud rate on the LSC.

PDMA0 - Priority bit for DMA Channel 0 interrupt, see IPN1.

PDMA1 - Priority bit for DMA Channel 1 interrupt, see IPN1.

PGSRE - Priority bit for GSC Receive Error interrupt, see IPN1.

PGSRV - Priority bit for GSC Receive Valid interrupt, see IPN1.

PGSTE - Priority bit for GSC Transmit Error interrupt, see IPN1.

PGSTV - Priority bit for GSC Transmit Valid interrupt, see IPN1.

PL0 - One of two bits that determines the Preamble Length, see GMOD.

PL1 - One of two bits that determines the Preamble Length, see GMOD.

PRBS - (0E4H) Pseudo-Random Binary Sequence, generates the pseudo-random number to be used in CSMA/CD backoff algorithms.

PS - Priority bit for the LSC service interrupt, see IP.

PT0 - Priority bit for Timer 0 interrupt, see IP.

PT1 - Priority bit for Timer 1 interrupt, see IP.

PX0 - Priority bit for External interrupt 0, see IP.

PX1 - Priority bit for External interrupt 1, see IP.

RCABT - GSC Receiver Abort error bit, see RSTAT.

RDN - GSC Receiver Done bit, see RSTAT.

GREN - GSC Receiver Enable bit, see RSTAT.

RFNE - GSC Receive FIFO Not Empty bit, see RSTAT.

RI - LSC Receive Interrupt bit, see SCON.

RFIFO - (F4H) RFIFO is a 3-byte FIFO that contains the receive data from the GSC.

RSTAT (0E8H) - Receive Status Register

7	6	5	4	3	2	1	0
OVR	RCABT	AE	CRCE	RDN	RFNE	GREN	HABEN

RSTAT.0 (HBAEN) - Hardware Based Acknowledge Enable - If set, enables the hardware based acknowledge feature.

RSTAT.1 (GREN) - Receiver Enable - When set, the receiver is enabled to accept incoming frames. This also clears RDN, CRCE, AE, RCABT and the receive FIFO. It is cleared by the receiver at the end of a reception or if any errors occurred. The status of GREN has no effect on whether the receiver detects a collision in CSMA/CD mode as the receiver input circuitry always monitors the receive pin.

RSTAT.2 (RFNE) - Receive FIFO Not Empty - If set, indicates that the receive FIFO contains data. The receive FIFO is a three byte buffer into which the receive data is loaded. A CPU read of the FIFO retrieves the oldest data and automatically updates the FIFO pointers. Setting GREN to a one will clear the receive FIFO. The status of this flag is controlled by the GSC. This bit is cleared if user S/W empties receive FIFO.

RSTAT.3 (RDN) - Receive Done - If set, indicates the successful completion of a receiver operation. Will not be set if a CRC, alignment, abort, or FIFO overrun error occurred.

RSTAT.4 (CRCE) - CRC Error - If set, indicates that a properly aligned frame was received with a mismatched CRC.

RSTAT.5 (AE) - Alignment Error - If set, indicates that the line went idle when the receiver shift register was not full and the resulting CRC was bad in the CSMA/CD mode. If a correct CRC was valid then AE is not set. In SDLC mode, AE indicates that a non-byte-aligned flag was received.

RSTAT.6 (RCABT) - Receiver Collision/Abort Detect - If set, indicates that a collision was detected after data had been loaded into the receive FIFO in CSMA/CD mode. In SDLC mode, RCABT indicates that 7 consecutive ones were detected prior to the end flag but after data has been loaded into the receive FIFO.

RSTAT.7 (OVR) - Overrun - If set, indicates that the receive FIFO was full and new shift register data was written into it. It is cleared by user S/W.

SARH0 (0A3H) - Source Address Register High 0, contains the high byte of the source address for DMA Channel 0.

SARH1 (0B3H) - Source Address Register High 1, contains the high byte of the source address for DMA Channel 1.

SARL0 (0A2H) - Source Address Register Low 0, contains the low byte of the source address for DMA Channel 0.

SARL1 (0B2H) - Source Address Register Low 1, contains the low byte of the source address for DMA Channel 1.

SAS - Source Address Space bit, see DCON0.

SBUF (099H) - Serial Buffer, both the receive and transmit SFR location for the LSC.

SCON (098H)

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SCON.0 (RI) - Receive Interrupt flag.

SCON.1 (TI) - Transmit Interrupt flag.

SCON.2 (RB8) - Receive Bit 8, contains the ninth bit that was received in Modes 2 and 3 or the stop bit in Mode 1 if SM20. Not used in Mode 0.

SCON.3 (TB8) - Transmit Bit 8, the ninth bit to be transmitted in Modes 2 and 3.

SCON.4 (REN) - Receiver Enable, enables reception for the LSC.

SCON.5 (SM2) - Enables the multiprocessor communication feature in Modes 2 and 3 for the LSC.

SCON.6 (SM1) - LSC mode specifier.

SCON.7 (SM2) - LSC mode specifier.

SDLC - Stands for Synchronous Data Link Communication and is a protocol developed by IBM.

SLOTTM - (0B4H) Determines the length of the slot time in CSMA/CD.

SP (081H) - Stack Pointer, an eight bit pointer register used during a PUSH, POP, CALL, RET, or RETI.

TCDCNT - (0D4H) Contains the number of collisions in the current frame if using probabilistic CSMA/CD and contains the maximum number of slots in the deterministic mode.

TCDT - Transmit Collision Detect, see TSTAT.

TCON (088H)

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TCON.0 (IT0) - Interrupt 0 mode control bit.

TCON.1 (IE0) - External interrupt 0 edge flag.

TCON.2 (IT1) - Interrupt 1 mode control bit.

TCON.3 (IE1) - External interrupt 1 edge flag.

TCON.4 (TR0) - Timer 0 run control bit.

CON.5 (TF0) - Timer 0 overflow flag.

TCON.6 (TR1) - Timer 1 run control bit.

TCON.7 (TF1) - Timer 1 overflow flag.

TDN - Transmit Done flag, see TSTAT.

TEN - Transmit Enable bit, see TSTAT.

TFNF - Transmit FIFO Not Full flag, see TSTAT.

TFIFO - (85H) TFIFO is a 3-byte FIFO that contains the transmission data for the GSC.

TH0 (08CH) - Timer 0 High byte, contains the high byte for timer/counter 0.

TH1 (08DH) - Timer 1 High byte, contains the high byte for timer/counter 1.

TI - Transmit Interrupt, see SCON.

TL0 (08AH) - Timer 0 Low byte, contains the low byte for timer/counter 0.

TL1 (08BH) - Timer 1 Low byte, contains the low byte for timer/counter 1.

TM - Transfer Mode, see, DCON0.

TMOD (089H)

7	6	5	4	3	2	1	0
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0

TMOD.0 (M0) - Mode selector bit for Timer 0.

TMOD.1 (M1) - Mode selector bit for Timer 0.

TMOD.2 (C/$\overline{T}$) - Timer/Counter selector bit for Timer 0.

TMOD.3 (GATE) - Gating Mode bit for Timer 0.

TMOD.4 (M0) - Mode selector bit for Timer 1.

TMOD.5 (M1) - Mode selector bit for Timer 1.

TMOD.6 (C/$\overline{T}$) - Timer/Counter selector bit for Timer 1.

TMOD.7 (GATE) - Gating Mode bit for Timer 1.

TSTAT (0D8) - Transmit Status Register

7	6	5	4	3	2	1	0
LNI	NOACK	UR	TCDT	TDN	TFNF	TEN	DMA

TSTAT.0 (DMA) - DMA Select - If set, indicates that DMA channels are used to service the GSC FIFO's and GSC interrupts occur on TDN and RDN, and also enables UR to become set. If cleared, indicates that the GSC is operating in it normal mode and interrupts occur on TFNE and RFNE.For more information on DMA servicing please refer to the DMA section on DMA serial demand mode (4.2.2.3).

TSTAT.1 (TEN) - Transmit Enable - When set causes TDN, UR, TCDT, and NOACK flags to be reset and the TFIFO cleared. The transmitter will clear TEN after a successful transmission, a collision during the data, CRC, or end flag. If cleared during a transmission the GSC transmit pin goes to a steady state high level. This is the method used to send an abort character in SDLC. Also $\overline{DEN}$ is forced to a high level. The end of transmission is occurs whenever the TFIFO is emptied.

TSTAT.2 (TFNF) - Transmit FIFO not full - When set, indicates that new data may be written into the transmit FIFO. The transmit FIFO is a three byte buffer that loads the transmit shift register with data.

TSTAT.3 (TDN) - Transmit Done - When set, indicates the successful completion of a frame transmission. If HBAEN is set, TDN will not be set until the end of the IFS following the transmitted message, so that the acknowledge can be checked. If an acknowledge is expected and not received, TDN is not set. An acknowledge is not expected following a broadcast or multi-cast packet.

TSTAT.4 (TCDT) - Transmit Collision Detect - If set, indicates that the transmitter halted due to a collision. It is set if a collision occurs during the data or CRC or if there are more than eight collisions.

TSTAT.5 (UR) - Underrun - If set, indicates that in DMA mode the last bit was shifted out of the transmit register and that the DMA byte count did not equal zero. When an underrun occurs, the transmitter halts without sending the CRC or the end flag.

TSTAT.6 (NOACK) - No Acknowledge - If set, indicates that no acknowledge was received for the previous frame. Will be set only if HBAEN is set and no acknowledge is received prior to the end of the IFS. NOACK is not set following a broadcast or a multicast packet.

TSTAT.7 (LNI) - Line Idle - If set, indicates the receive line is idle. In SDLC protocol it is set if 15 consecutive ones are received. In CSMA/CD protocol, line idle is set if no transitions occur on GR$\times$D for 1.6 bit times after a required transition. LNI is cleared after a transition on GR$\times$D.

TxC - External Clock input for GSC transmitter.

UR - Underrun flag, see TSTAT.

XRCLK - External GSC Receive Clock Enable bit, see PCON.

XTCLK - External GSC Transmit Clock Enable bit, see GMOD.

MCS®-51 Programmer's Guide and Instruction Set

9

MCS®-51 PROGRAMMER'S GUIDE AND INSTRUCTION SET

The information presented in this chapter is collected from the previous MCS®-51 chapters of this book. The material has been selected and rearranged to form a quick and convenient reference for the programmers of the MCS-51. This guide pertains specifically to the 8051, 8052 and 80C51.

The following list should make it easier to find a subject in this chapter.

Memory Organization
- **Program Memory** 9-2
- **Data Memory** 9-3
- **Direct and Indirect Address Area** 9-5

Special Function Registers 9-7
Contents of SFRs after Power-On 9-8
SFR Memory Map 9-9
Program Status Word (PSW) 9-10
Power Control Register (PCON) 9-10
Interrupts 9-11
Interrupt Enable Register (IE) 9-11
Assigning Priority Level 9-12
Interrupt Priority Register 9-12
Timer/Counter Control Register (TCON) 9-13
Timer/Counter Mode Control Register (TMOD) 9-13
Timer Set-Up 9-14
- **Timer/Counter 0** 9-14
- **Timer/Counter 1** 9-15

Timer/Counter 2 Control Register (T2CON) 9-16
Timer/Counter 2 Set-Up 9-17
Serial Port Control Register 9-18
Serial Port Set-Up 9-18
Generating Baud Rates 9-19
MCS-51 Instruction Set 9-20
Instruction Definitions 9-24

MEMORY ORGANIZATION

PROGRAM MEMORY

The 8051 has separate address spaces for Program Memory and Data Memory. The Program Memory can be up to 64K bytes long. The lower 4K (8K for the 8052) may reside on-chip.

Figure 1 shows a map of the 8051 program memory, and Figure 2 shows a map of the 8052 program memory.

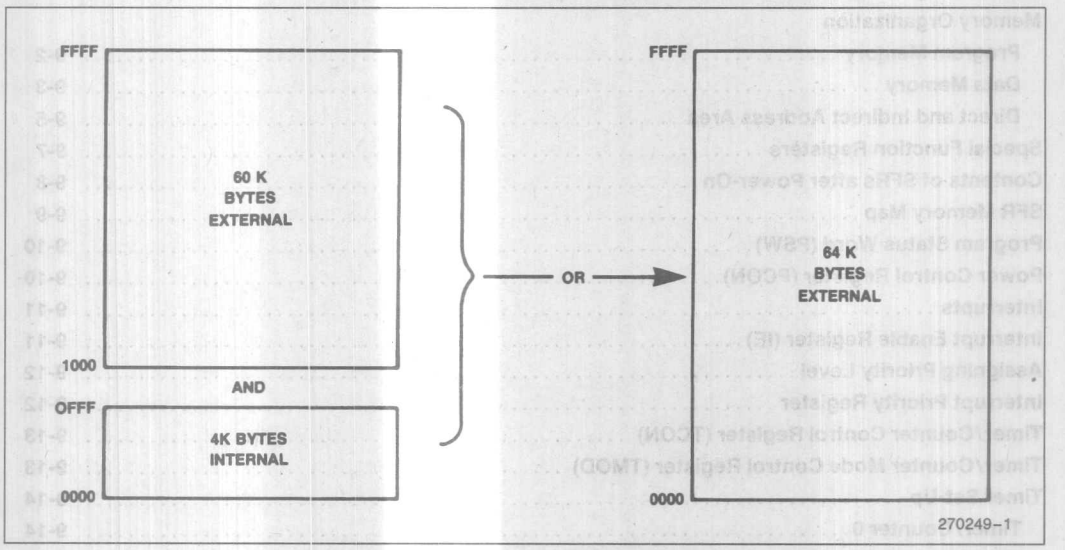


Figure 1. The 8051 Program Memory

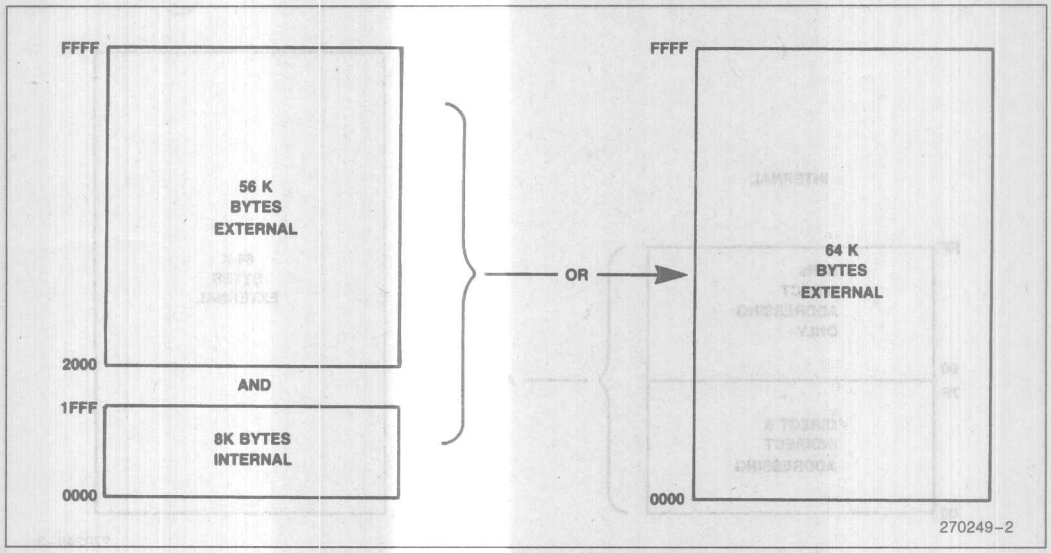


Figure 2. The 8052 Program Memory

Data Memory:

The 8051 can address up to 64K bytes of Data Memory to the chip. The "MOVX" instruction is used to access the external data memory. (Refer to the MCS-51 Instruction Set, in this chapter, for detailed description of instructions).

The 8051 has 128 bytes of on-chip RAM (256 bytes in the 8052) plus a number of Special Function Registers (SFRs). The lower 128 bytes of RAM can be accessed either by direct addressing (MOV data addr) or by indirect addressing (MOV @Ri). Figure 3 shows the 8051 and the 8052 Data Memory organization.

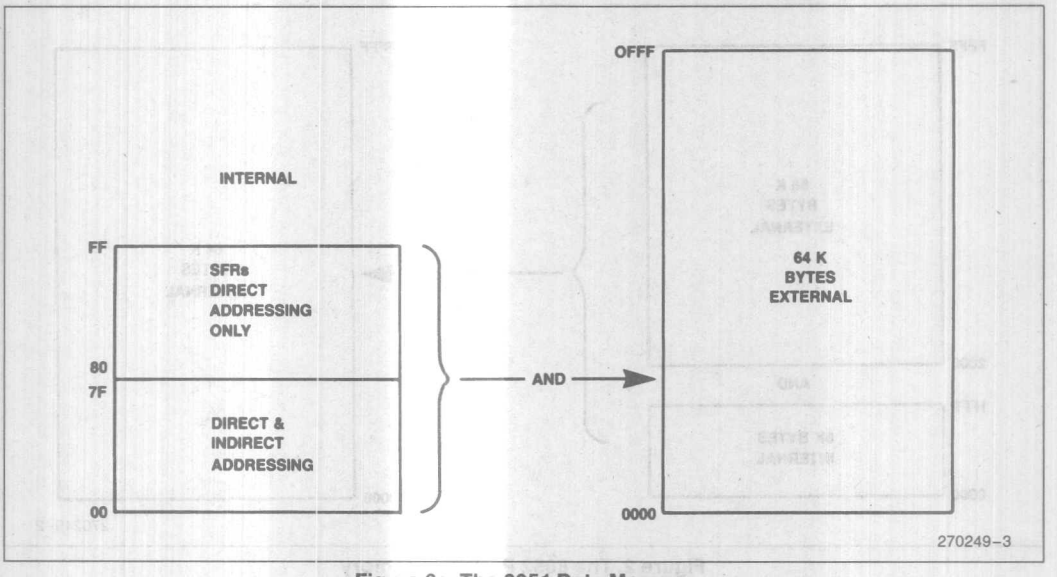


Figure 3a. The 8051 Data Memory

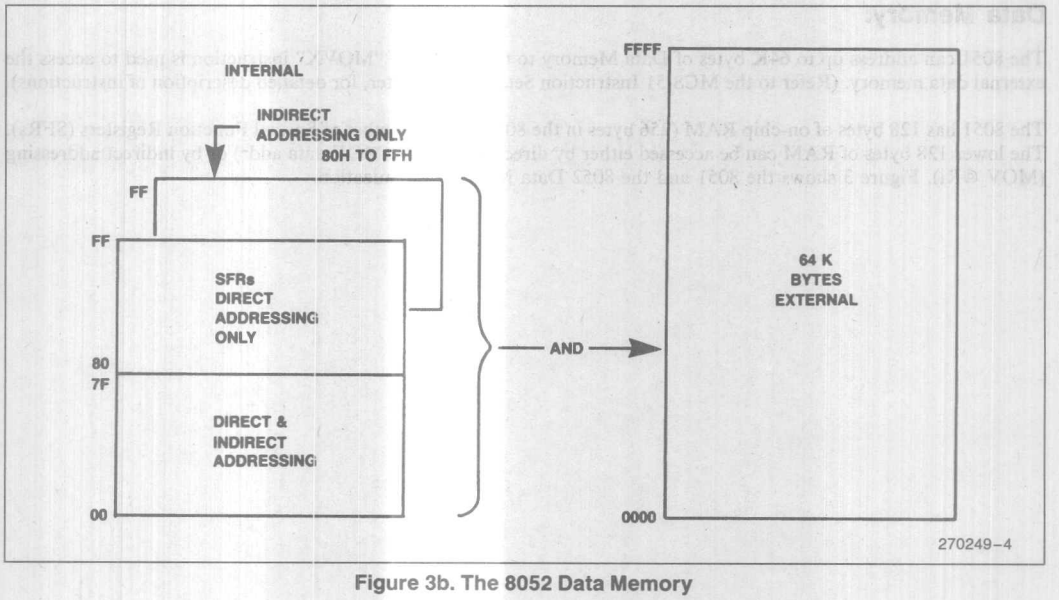


Figure 3b. The 8052 Data Memory

INDIRECT ADDRESS AREA:

Note that in Figure 3b the SFRs and the indirect address RAM have the same addresses (80H–0FFH). Nevertheless, they are two separate areas and are accessed in two different ways.

For example the instruction

```
MOV    80H,#0AAH
```

writes 0AAH to Port 0 which is one of the SFRs and the instruction

```
MOV    R0,#80H

MOV    @R0,#0BBH
```

writes 0BBH in location 80H of the data RAM. Thus, after execution of both of the above instructions Port 0 will contain 0AAH and location 80 of the RAM will contain 0BBH.

DIRECT AND INDIRECT ADDRESS AREA:

The 128 bytes of RAM which can be accessed by both direct and indirect addressing can be divided into 3 segments as listed below and shown in Figure 4.

1. Register Banks 0-3: Locations 0 through 1FH (32 bytes). ASM-51 and the device after reset default to register bank 0. To use the other register banks the user must select them in the software (refer to the MCS-51 Micro Assembler User's Guide). Each register bank contains 8 one-byte registers, 0 through 7.

Reset initializes the Stack Pointer to location 07H and it is incremented once to start from location 08H which is the first register (RO) of the second register bank. Thus, in order to use more than one register bank, the SP should be intialized to a different location of the RAM where it is not used for data storage (ie, higher part of the RAM).

2. Bit Addressable Area: 16 bytes have been assigned for this segment, 20H-2FH. Each one of the 128 bits of this segment can be directly addressed (0-7FH).

The bits can be referred to in two ways both of which are acceptable by the ASM-51. One way is to refer to their addresses, ie. 0 to 7FH. The other way is with reference to bytes 20H to 2FH. Thus, bits 0–7 can also be referred to as bits 20.0–20.7, and bits 8-FH are the same as 21.0–21.7 and so on.

Each of the 16 bytes in this segment can also be addressed as a byte.

3. Scratch Pad Area: Bytes 30H through 7FH are available to the user as data RAM. However, if the stack pointer has been initialized to this area, enough number of bytes should be left aside to prevent SP data destruction.

Figure 4 shows the different segments of the on-chip RAM.

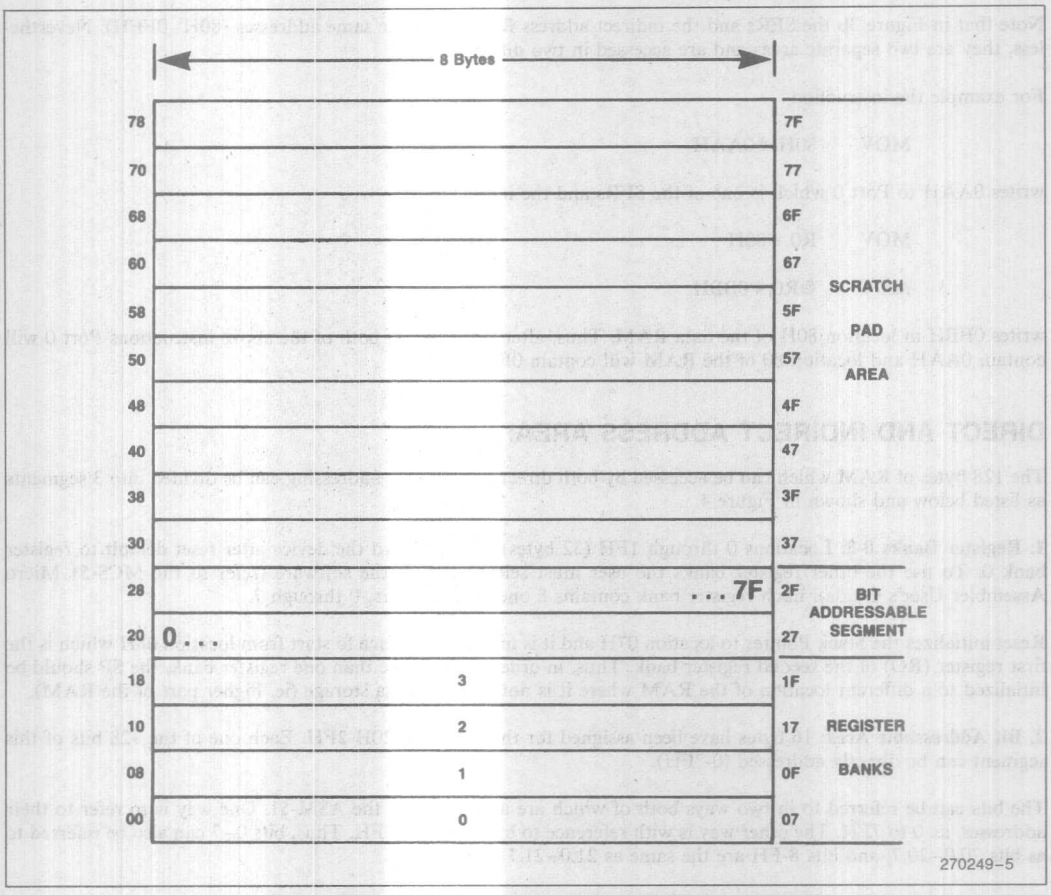


Figure 4. 128 Bytes of RAM Direct and Indirect Addressable

SPECIAL FUNCTION REGISTERS:

Table 1 contains a list of all the SFRs and their addresses.

Comparing Table 1 and Figure 5 shows that all of the SFRs that are byte and bit addressable are located on the first column of the diagram in Figure 5.

Table 1

Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
DPL	Low Byte	82H
DPH	High Byte	83H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
*TCON	Timer/Counter Control	88H
*+T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
+TH2	Timer/Counter 2 High Byte	0CDH
+TL2	Timer/Counter 2 Low Byte	0CCH
+RCAP2H	T/C 2 Capture Reg. High Byte	0CBH
+RCAP2L	T/C 2 Capture Reg. Low Byte	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

* = Bit addressable
+ = 8052 only

WHAT DO THE SFRs CONTAIN JUST AFTER POWER-ON OR A RESET?

Table 2 lists the contents of each SFR after power-on or a hardware reset.

Table 2. Contents of the SFRs after reset

Register	Value in Binary
*ACC	00000000
*B	00000000
*PSW	00000000
SP	00000111
DPTR	
DPH	00000000
DPL	00000000
*P0	11111111
*P1	11111111
*P2	11111111
*P3	11111111
*IP	8051 XXX00000, 8052 XX000000
*IE	8051 0XX00000, 8052 0X000000
TMOD	00000000
*TCON	00000000
*+T2CON	00000000
TH0	00000000
TL0	00000000
TH1	00000000
TL1	00000000
+TH2	00000000
+TL2	00000000
+RCAP2H	00000000
+RCAP2L	00000000
*SCON	00000000
SBUF	Indeterminate
PCON	HMOS 0XXXXXXX CHMOS 0XXX0000

X = Undefined
* = Bit Addressable
\+ = 8052 only

SFR MEMORY MAP

8 Bytes

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

↑
Bit
Addressable

Figure 5

Those SFRs that have their bits assigned for various functions are listed in this section. A brief description of each bit is provided for quick reference. For more detailed information refer to the Architecture Chapter of this book.

PSW: PROGRAM STATUS WORD. BIT ADDRESSABLE.

CY	AC	F0	RS1	RS0	OV	—	P

CY	PSW.7	Carry Flag.
AC	PSW.6	Auxiliary Carry Flag.
F0	PSW.5	Flag 0 available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1 (SEE NOTE 1).
RS0	PSW.3	Register Bank selector bit 0 (SEE NOTE 1).
OV	PSW.2	Overflow Flag.
—	PSW.1	Not implemented, reserved for future use.*
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of '1' bits in the accumulator.

NOTE:
1. The value presented by RS0 and RS1 selects the corresponding register bank.

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

PCON: POWER CONTROL REGISTER. NOT BIT ADDRESSABLE.

SMOD	—	—	—	GF1	GF0	PD	IDL

SMOD	Double baud rate bit. If Timer 1 is used to generate baud rate and SMOD = 1, the baud rate is doubled when the Serial Port is used in modes 1, 2, or 3.
—	Not implemented, reserved for future use.*
—	Not implemented, reserved for future use.*
—	Not implemented, reserved for future use.*
GF1	General purpose flag bit.
GF0	General purpose flag bit.
PD	Power Down bit. Setting this bit activates Power Down operation in the 80C51BH. (Available only in CHMOS).
IDL	Idle Mode bit. Setting this bit activates Idle Mode operation in the 80C51BH. (Available only in CHMOS).

If 1s are written to PD and IDL at the same time, PD takes precedence.

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

INTERRUPTS:

In order to use any of the interrupts in the MCS-51, the following three steps must be taken.

1. Set the EA (enable all) bit in the IE register to 1.
2. Set the corresponding individual interrupt enable bit in the IE register to 1.
3. Begin the interrupt service routine at the corresponding Vector Address of that interrupt. See Table below.

Interrupt Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI & TI	0023H
TF2 & EXF2	002BH

In addition, for external interrupts, pins $\overline{INT0}$ and $\overline{INT1}$ (P3.2 and P3.3) must be set to 1, and depending on whether the interrupt is to be level or transition activated, bits IT0 or IT1 in the TCON register may need to be set to 1.

ITx = 0 level activated

ITx = 1 transition activated

IE: INTERRUPT ENABLE REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

EA	—	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	Not implemented, reserved for future use.*
ET2	IE.5	Enable or disable the Timer 2 overflow or capture interrupt (8052 only).
ES	IE.4	Enable or disable the serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External Interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

ASSIGNING HIGHER PRIORITY TO ONE OR MORE INTERRUPTS:

In order to assign higher priority to an interrupt the corresponding bit in the IP register must be set to 1.

Remember that while an interrupt service is in progress, it cannot be interrupted by a lower or same level interrupt.

PRIORITY WITHIN LEVEL:

Priority within level is only to resolve simultaneous requests of the same priority level.

From high to low, interrupt sources are listed below:

IE0
TF0
IE1
TF1
RI or TI
TF2 or EXF2

IP: INTERRUPT PRIORITY REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt has a lower priority and if the bit is 1 the corresponding interrupt has a higher priority.

—	—	PT2	PS	PT1	PX1	PT0	PX0

—	IP. 7	Not implemented, reserved for future use.*
—	IP. 6	Not implemented, reserved for future use.*
PT2	IP. 5	Defines the Timer 2 interrupt priority level (8052 only).
PS	IP. 4	Defines the Serial Port interrupt priority level.
PT1	IP. 3	Defines the Timer 1 interrupt priority level.
PX1	IP. 2	Defines External Interrupt 1 priority level.
PT0	IP. 1	Defines the Timer 0 interrupt priority level.
PX0	IP. 0	Defines the External Interrupt 0 priority level.

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

TCON: TIMER/COUNTER CONTROL REGISTER. BIT ADDRESSABLE.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 TCON. 7 Timer 1 overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.

TR1 TCON. 6 Timer 1 run control bit. Set/cleared by software to turn Timer/Counter 1 ON/OFF.

TF0 TCON. 5 Timer 0 overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the service routine.

TR0 TCON. 4 Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.

IE1 TCON. 3 External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.

IT1 TCON. 2 Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

IE0 TCON. 1 External Interrupt 0 edge flag. Set by hardware when External Interrupt edge detected. Cleared by hardware when interrupt is processed.

IT0 TCON. 0 Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

TMOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE.

GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
TIMER 1				TIMER 0			

GATE When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).

C/$\overline{T}$ Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).

M1 Mode selector bit. (NOTE 1)

M0 Mode selector bit. (NOTE 1)

NOTE 1:

M1	M0		Operating Mode
0	0	0	13-bit Timer (MCS-48 compatible)
0	1	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits, TH0 is an 8-bit Timer and is controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter 1 stopped.

TIMER SET-UP

Tables 3 through 6 give some values for TMOD which can be used to set up Timer 0 in different modes.

It is assumed that only one timer is being used at a time. If it is desired to run Timers 0 and 1 simultaneously, in any mode, the value in TMOD for Timer 0 must be ORed with the value shown for Timer 1 (Tables 5 and 6).

For example, if it is desired to run Timer 0 in mode 1 GATE (external control), and Timer 1 in mode 2 COUNTER, then the value that must be loaded into TMOD is 69H (09H from Table 3 ORed with 60H from Table 6).

Moreover, it is assumed that the user, at this point, is not ready to turn the timers on and will do that at a different point in the program by setting bit TRx (in TCON) to 1.

TIMER/COUNTER 0

As a Timer:

Table 3

MODE	TIMER 0 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	08H
1	16-bit Timer	01H	09H
2	8-bit Auto-Reload	02H	0AH
3	two 8-bit Timers	03H	0BH

As a Counter:

Table 4

MODE	COUNTER 0 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	04H	0CH
1	16-bit Timer	05H	0DH
2	8-bit Auto-Reload	06H	0EH
3	one 8-bit Counter	07H	0FH

NOTES:
1. The Timer is turned ON/OFF by setting/clearing bit TR0 in the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on $\overline{INT0}$ (P3.2) when TR0 = 1 (hardware control).

TIMER/COUNTER 1

As a Timer:

Table 5

MODE	TIMER 1 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	80H
1	16-bit Timer	10H	90H
2	8-bit Auto-Reload	20H	A0H
3	does not run	30H	B0H

As a Counter:

Table 6

MODE	COUNTER 1 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	40H	C0H
1	16-bit Timer	50H	D0H
2	8-bit Auto-Reload	60H	E0H
3	not available	—	—

NOTES:
1. The Timer is turned ON/OFF by setting/clearing bit TR1 in the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on $\overline{INT1}$ (P3.3) when TR1 = 1 (hardware control).

T2CON: TIMER/COUNTER 2 CONTROL REGISTER. BIT ADDRESSABLE

8052 Only

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

TF2 T2CON. 7 Timer 2 overflow flag set by hardware and cleared by software. TF2 cannot be set when either RCLK = 1 or CLK = 1

EXF2 T2CON. 6 Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX, and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.

RCLK T2CON. 5 Receive clock flag. When set, causes the Serial Port to use Timer 2 overflow pulses for its receive clock in modes 1 & 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.

TLCK T2CON. 4 Transmit clock flag. When set, causes the Serial Port to use Timer 2 overflow pulses for its transmit clock in modes 1 & 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.

EXEN2 T2CON. 3 Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of negative transition on T2EX if Timer 2 is not being used to clock the Serial Port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.

TR2 T2CON. 2 Software START/STOP control for Timer 2. A logic 1 starts the Timer.

C/$\overline{T2}$ T2CON. 1 Timer or Counter select.

0 = Internal Timer. 1 = External Event Counter (falling edge triggered).

CP/$\overline{RL2}$ T2CON. 0 Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, Auto-Reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the Timer is forced to Auto-Reload on Timer 2 overflow.

TIMER/COUNTER 2 SET-UP

Except for the baud rate generator mode, the values given for T2CON do not include the setting of the TR2 bit. Therefore, bit TR2 must be set, separately, to turn the Timer on.

As a Timer:

Table 7

MODE	T2CON	
	INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
16-bit Auto-Reload	00H	08H
16-bit Capture	01H	09H
BAUD rate generator receive & transmit same baud rate	34H	36H
receive only	24H	26H
transmit only	14H	16H

As a Counter:

Table 8

MODE	TMOD	
	INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
16-bit Auto-Reload	02H	0AH
16-bit Capture	03H	0BH

NOTES:
1. Capture/Reload occurs only on Timer/Counter overflow.
2. Capture/Reload occurs on Timer/Counter overflow and a 1 to 0 transition on T2EX (P1.1) pin except when Timer 2 is used in the baud rate generating mode.

SCON: SERIAL PORT CONTROL REGISTER. BIT ADDRESSABLE.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 SCON. 7 Serial Port mode specifier. (NOTE 1).

SM1 SCON. 6 Serial Port mode specifier. (NOTE 1).

SM2 SCON. 5 Enables the multiprocessor communication feature in modes 2 & 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0. (See Table 9).

REN SCON. 4 Set/Cleared by software to Enable/Disable reception.

TB8 SCON. 3 The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.

RB8 SCON. 2 In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.

TI SCON. 1 Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.

RI SCON. 0 Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes (except see SM2). Must be cleared by software.

NOTE 1:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	SHIFT REGISTER	Fosc./12
0	1	1	8-Bit UART	Variable
1	0	2	9-Bit UART	Fosc./64 OR Fosc./32
1	1	3	9-Bit UART	Variable

SERIAL PORT SET-UP:

Table 9

MODE	SCON	SM2 VARIATION
0	10H	Single Processor Environment (SM2 = 0)
1	50H	
2	90H	
3	D0H	
0	NA	Multiprocessor Environment (SM2 = 1)
1	70H	
2	B0H	
3	F0H	

GENERATING BAUD RATES

Serial Port in Mode 0:

Mode 0 has a fixed baud rate which is 1/12 of the oscillator frequency. To run the serial port in this mode none of the Timer/Counters need to be set up. Only the SCON register needs to be defined.

$$\text{Baud Rate} = \frac{\text{Osc Freq}}{12}$$

Serial Port in Mode 1:

Mode 1 has a variable baud rate. The baud rate can be generated by either Timer 1 or Timer 2 (8052 only).

USING TIMER/COUNTER 1 TO GENERATE BAUD RATES:

For this purpose, Timer 1 is used in mode 2 (Auto-Reload). Refer to Timer Setup section of this chapter.

$$\text{Baud Rate} = \frac{\text{K x Oscillator Freq.}}{32 \times 12 \times [256 - (\text{TH1})]}$$

If SMOD = 0, then K = 1.
If SMOD = 1, then K = 2. (SMOD is the PCON register).

Most of the time the user knows the baud rate and needs to know the reload value for TH1. Therefore, the equation to calculate TH1 can be written as:

$$\text{TH1} = 256 - \frac{\text{K x Osc Freq.}}{384 \times \text{baud rate}}$$

TH1 must be an integer value. Rounding off TH1 to the nearest integer may not produce the desired baud rate. In this case, the user may have to choose another crystal frequency.

Since the PCON register is not bit addressable, one way to set the bit is logical ORing the PCON register. (ie, ORL PCON,#80H). The address of PCON is 87H.

USING TIMER/COUNTER 2 TO GENERATE BAUD RATES:

For this purpose, Timer 2 must be used in the baud rate generating mode. Refer to Timer 2 Setup Table in this chapter. If Timer 2 is being clocked through pin T2 (P1.0) the baud rate is:

$$\text{Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

And if it is being clocked internally the baud rate is:

$$\text{Baud Rate} = \frac{\text{Osc Freq}}{32 \times [65536 - (\text{RCAP2H, RCAP2L})]}$$

To obtain the reload value for RCAP2H and RCAP2L the above equation can be rewritten as:

$$\text{RCAP2H, RCAP2L} = 65536 - \frac{\text{Osc Freq}}{32 \times \text{Baud Rate}}$$

SERIAL PORT IN MODE 2:

The baud rate is fixed in this mode and is $1/32$ or $1/64$ of the oscillator frequency depending on the value of the SMOD bit in the PCON register.

In this mode none of the Timers are used and the clock comes from the internal phase 2 clock.

SMOD = 1, Baud Rate = $1/32$ Osc Freq.

SMOD = 0, Baud Rate = $1/64$ Osc Freq.

To set the SMOD bit: ORL PCON,#80H. The address of PCON is 87H.

SERIAL PORT IN MODE 3:

The baud rate in mode 3 is variable and sets up exactly the same as in mode 1.

MCS®-51 INSTRUCTION SET

Table 10. 8051 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings(1)

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,/bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

(1)Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on instruction set and addressing modes:

Rn — Register R7–R0 of the currently selected Register Bank.

direct — 8-bit internal data location's address. This could be an Internal Data RAM location (0–127) or a SFR [i.e., I/O port, control register, status register, etc. (128–255)].

@Ri — 8-bit internal data RAM location (0–255) addressed indirectly through register R1 or R0.

#data — 8-bit constant included in instruction.

#data 16 — 16-bit constant included in instruction.

addr 16 — 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.

addr 11 — 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.

rel — Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to +127 bytes relative to first byte of the following instruction.

bit — Direct Addressed bit in Internal Data RAM or Special Function Register.

* — New operation not provided by 8048AH/8049AH.

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,Rn	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12

All mnemonics copyrighted ©Intel Corporation 1980

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS (Continued)				
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A,Rn	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS (Continued)				
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	Rn,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12

All mnemonics copyrighted ©Intel Corporation 1980

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER (Continued)				
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri, #data	Move immediate data to indirect RAM	2	12
MOV	DPTR, #data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX	@Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12

Mnemonic		Description	Byte	Oscillator Period
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24

All mnemonics copyrighted ©Intel Corporation 1980

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				
JMP	@A + DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A, # data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE	Rn, # data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@Ri, # data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

All mnemonics copyrighted © Intel Corporation 1980

INSTRUCTION DEFINITIONS

ACALL addr11

Function: Absolute Call

Description: ACALL unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction, then pushes the 16-bit result onto the stack (low-order byte first) and increments the Stack Pointer twice. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, opcode bits 7-5, and the second byte of the instruction. The subroutine called must therefore start within the same 2K block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

Example: Initially SP equals 07H. The label "SUBRTN" is at program memory location 0345 H. After executing the instruction,

```
ACALL   SUBRTN
```

at location 0123H, SP will contain 09H, internal RAM locations 08H and 09H will contain 25H and 01H, respectively, and the PC will contain 0345H.

Bytes: 2

Cycles: 2

Encoding:

a10 a9 a8 1	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Operation:

ACALL
(PC) ← (PC) + 2
(SP) ← (SP) + 1
((SP)) ← (PC_{7-0})
(SP) ← (SP) + 1
((SP)) ← (PC_{15-8})
(PC_{10-0}) ← page address

ADD A,<src-byte>

Function: Add

Description: ADD adds the byte variable indicated to the Accumulator, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occured.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B). The instruction,

ADD A,R0

will leave 6DH (01101101B) in the Accumulator with the AC flag cleared and both the carry flag and OV set to 1.

ADD A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 1 0	1 r r r

Operation: ADD
(A) ← (A) + (Rn)

ADD A,direct

Bytes: 2

Cycles: 1

Encoding:

0 0 1 0	0 1 0 1	direct address

Operation: ADD
(A) ← (A) + (direct)

ADD A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 1 0	0 1 1 i

Operation: ADD
(A) ← (A) + ((Ri))

ADD A,#data

Bytes: 2

Cycles: 1

Encoding:

0 0 1 0	0 1 0 0	immediate data

Operation: ADD
(A) ← (A) + #data

ADDC A,<src-byte>

Function: Add with Carry

Description: ADDC simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occured.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) with the carry flag set. The instruction,

ADDC A,R0

will leave 6EH (01101110B) in the Accumulator with AC cleared and both the Carry flag and OV set to 1.

ADDC A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 1 1	1 r r r

Operation: ADDC
(A) ← (A) + (C) + (R_n)

ADDC A,direct

Bytes: 2

Cycles: 1

Encoding:

0 0 1 1	0 1 0 1	direct address

Operation: ADDC
(A) ← (A) + (C) + (direct)

ADDC A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 1 1	0 1 1 i

Operation: ADDC
(A) ← (A) + (C) + ((R_i))

ADDC A,#data

Bytes: 2

Cycles: 1

Encoding:

0 0 1 1	0 1 0 0	immediate data

Operation: ADDC
(A) ← (A) + (C) + #data

AJMP addr11

Function: Absolute Jump

Description: AJMP transfers program execution to the indicated address, which is formed at run-time by concatenating the high-order five bits of the PC (*after* incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must therefore be within the same 2K block of program memory as the first byte of the instruction following AJMP.

Example: The label "JMPADR" is at program memory location 0123H. The instruction,

AJMP JMPADR

is at location 0345H and will load the PC with 0123H.

Bytes: 2

Cycles: 2

Encoding:

a10 a9 a8 0	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Operation: AJMP
(PC) ← (PC) + 2
($PC_{10\text{-}0}$) ← page address

ANL <dest-byte>,<src-byte>

Function: Logical-AND for byte variables

Description: ANL performs the bitwise logical-AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: If the Accumulator holds 0C3H (11000011B) and register 0 holds 55H (01010101B) then the instruction,

ANL A,R0

will leave 41H (01000001B) in the Accumulator.

When the destination is a directly addressed byte, this instruction will clear combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be a constant contained in the instruction or a value computed in the Accumulator at run-time. The instruction,

ANL P1,#01110011B

will clear bits 7, 3, and 2 of output port 1.

ANL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 0 1	1 r r r

Operation: ANL
(A) ← (A) ∧ (Rn)

ANL A,direct

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 1 0 1	direct address

Operation: ANL

(A) ← (A) ∧ (direct)

ANL A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 1 0 1	0 1 1 i

Operation: ANL

(A) ← (A) ∧ ((Ri))

ANL A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 1 0 0	immediate data

Operation: ANL
(A) ← (A) ∧ #data

ANL direct,A

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 0 1 0	direct address

Operation: ANL
(direct) ← (direct) ∧ (A)

ANL direct,#data

Bytes: 3

Cycles: 2

Encoding:

0 1 0 1	0 0 1 1	direct address	immediate data

Operation: ANL
(direct) ← (direct) ∧ #data

ANL C,<src-bit>

Function: Logical-AND for bit variables

Description: If the Boolean value of the source bit is a logical 0 then clear the carry flag; otherwise leave the carry flag in its current state. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, *but the source bit itself is not affected.* No other flags are affected.

Only direct addressing is allowed for the source operand.

Example: Set the carry flag if, and only if, P1.0 = 1, ACC. 7 = 1, and OV = 0:

```
MOV  C,P1.0     ;LOAD CARRY WITH INPUT PIN STATE
ANL  C,ACC.7    ;AND CARRY WITH ACCUM. BIT 7
ANL  C,/OV      ;AND WITH INVERSE OF OVERFLOW FLAG
```

ANL C,bit

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 0 1 0	bit address

Operation: ANL
(C) ← (C) ∧ (bit)

ANL C,/bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 1	0 0 0 0	bit address

Operation: ANL
(C) ← (C) ∧ ¬(bit)

CJNE <dest-byte>,<src-byte>, rel

Function: Compare and Jump if Not Equal.

Description: CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The branch destination is computed by adding the signed relative-displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction. The carry flag is set if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations: the Accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

Example: The Accumulator contains 34H. Register 7 contains 56H. The first instruction in the sequence,

```
                CJNE   R7,#60H, NOT__EQ
;               ...    .....          ;  R7 = 60H.
NOT__EQ:        JC     REQ__LOW       ;  IF R7 < 60H.
;               ...    .....          ;  R7 > 60H.
```

sets the carry flag and branches to the instruction at label NOT__EQ. By testing the carry flag, this instruction determines whether R7 is greater or less than 60H.

If the data being presented to Port 1 is also 34H, then the instruction,

```
WAIT:  CJNE  A,P1,WAIT
```

clears the carry flag and continues with the next instruction in sequence, since the Accumulator does equal the data read from P1. (If some other value was being input on P1, the program will loop at this point until the P1 data changes to 34H.)

CJNE A,direct,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	0 1 0 1	direct address	rel. address

Operation:

```
(PC) ← (PC) + 3
IF (A) <> (direct)
THEN
        (PC) ← (PC) + relative offset

IF (A) < (direct)
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

CJNE A,#data,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	0 1 0 0	immediate data	rel. address

Operation:

```
(PC) ← (PC) + 3
IF (A) <> data
THEN
        (PC) ← (PC) + relative offset

IF (A) < data
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

CJNE Rn,#data,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	1 r r r	immediate data	rel. address

Operation:

```
(PC) ← (PC) + 3
IF (Rn) <> data
THEN
        (PC) ← (PC) + relative offset

IF (Rn) < data
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

CJNE @Ri,#data,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	0 1 1 i	immediate data	rel. address

Operation:

```
(PC) ← (PC) + 3
IF ((Ri)) <> data
THEN
        (PC) ← (PC) + relative offset

IF ((Ri)) < data
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

CLR A

Function: Clear Accumulator

Description: The Accumulator is cleared (all bits set on zero). No flags are affected.

Example: The Accumulator contains 5CH (01011100B). The instruction,

```
CLR A
```

will leave the Accumulator set to 00H (00000000B).

Bytes: 1

Cycles: 1

Encoding:

1 1 1 1	0 1 0 0

Operation:

```
CLR
(A) ← 0
```

CLR bit

Function: Clear bit

Description: The indicated bit is cleared (reset to zero). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

Example: Port 1 has previously been written with 5DH (01011101B). The instruction,

```
CLR P1.2
```

will leave the port set to 59H (01011001B).

CLR C

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	0 0 1 1

Operation:

```
CLR
(C) ← 0
```

CLR bit

Bytes: 2

Cycles: 1

Encoding:

1 1 0 0	0 0 1 0	bit address

Operation:

```
CLR
(bit) ← 0
```

CPL A

Function: Complement Accumulator

Description: Each bit of the Accumulator is logically complemented (one's complement). Bits which previously contained a one are changed to a zero and vice-versa. No flags are affected.

Example: The Accumulator contains 5CH (01011100B). The instruction,

```
CPL   A
```

will leave the Accumulator set to 0A3H (10100011B).

Bytes: 1

Cycles: 1

Encoding:

1 1 1 1	0 1 0 0

Operation:

```
CPL
(A) ← ¬(A)
```

CPL bit

Function: Complement bit

Description: The bit variable specified is complemented. A bit which had been a one is changed to zero and vice-versa. No other flags are affected. CLR can operate on the carry or any directly addressable bit.

Note: When this instruction is used to modify an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

Example: Port 1 has previously been written with 5BH (01011101B). The instruction sequence,

```
CPL   P1.1
CPL   P1.2
```

will leave the port set to 5BH (01011011B).

CPL C

Bytes: 1

Cycles: 1

Encoding:

1 0 1 1	0 0 1 1

Operation:

```
CPL
(C) ← ¬(C)
```

CPL bit

Bytes: 2

Cycles: 1

Encoding:

1 0 1 1	0 0 1 0	bit address

Operation: CPL
(bit) ← ¬ (bit)

DA A

Function: Decimal-adjust Accumulator for Addition

Description: DA A adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variables (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If Accumulator bits 3-0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag is one, six is added to the Accumulator producing the proper BCD digit in the low-order nibble. This internal addition would set the carry flag if a carry-out of the low-order four-bit field propagated through all high-order bits, but it would not clear the carry flag otherwise.

If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxxx-111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this would set the carry flag if there was a carry-out of the high-order bits, but wouldn't clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00H, 06H, 60H, or 66H to the Accumulator, depending on initial Accumulator and PSW conditions.

Note: DA A *cannot* simply convert a hexadecimal number in the Accumulator to BCD notation, nor does DA A apply to decimal subtraction.

Example: The Accumulator holds the value 56H (01010110B) representing the packed BCD digits of the decimal number 56. Register 3 contains the value 67H (01100111B) representing the packed BCD digits of the decimal number 67. The carry flag is set. The instruction sequence.

```
ADDC  A,R3
DA    A
```

will first perform a standard twos-complement binary addition, resulting in the value 0BEH (10111110) in the Accumulator. The carry and auxiliary carry flags will be cleared.

The Decimal Adjust instruction will then alter the Accumulator to the value 24H (00100100B), indicating the packed BCD digits of the decimal number 24, the low-order two digits of the decimal sum of 56, 67, and the carry-in. The carry flag will be set by the Decimal Adjust instruction, indicating that a decimal overflow occurred. The true sum 56, 67, and 1 is 124.

BCD variables can be incremented or decremented by adding 01H or 99H. If the Accumulator initially holds 30H (representing the digits of 30 decimal), then the instruction sequence,

```
ADD   A,#99H
DA    A
```

will leave the carry set and 29H in the Accumulator, since 30 + 99 = 129. The low-order byte of the sum can be interpreted to mean 30 − 1 = 29.

Bytes: 1

Cycles: 1

Encoding:

1 1 0 1	0 1 0 0

Operation: DA

-contents of Accumulator are BCD

IF [[(A_{3-0}) > 9] ∨ [(AC) = 1]]
THEN $(A_{3-0}) \leftarrow (A_{3-0}) + 6$
AND

IF [[(A_{7-4}) > 9] ∨ [(C) = 1]]
THEN $(A_{7-4}) \leftarrow (A_{7-4}) + 6$

DEC byte

Function: Decrement

Description: The variable indicated is decremented by 1. An original value of 00H will underflow to 0FFH. No flags are affected. Four operand addressing modes are allowed: accumulator, register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Register 0 contains 7FH (01111111B). Internal RAM locations 7EH and 7FH contain 00H and 40H, respectively. The instruction sequence,

```
DEC   @R0
DEC   R0
DEC   @R0
```

will leave register 0 set to 7EH and internal RAM locations 7EH and 7FH set to 0FFH and 3FH.

DEC A

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	0 1 0 0

Operation: DEC
(A) ← (A) − 1

DEC Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	1 r r r

Operation: DEC
(Rn) ← (Rn) − 1

DEC direct

Bytes: 2

Cycles: 1

Encoding:

0 0 0 1	0 1 0 1	direct address

Operation: DEC
(direct) ← (direct) − 1

DEC @Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	0 1 1 i

Operation: DEC
((Ri)) ← ((Ri)) − 1

DIV AB

Function: Divide

Description: DIV AB divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B. The Accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and OV flags will be cleared.

Exception: if B had originally contained 00H, the values returned in the Accumulator and B-register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

Example: The Accumulator contains 251 (0FBH or 11111011B) and B contains 18 (12H or 00010010B). The instruction,

DIV AB

will leave 13 in the Accumulator (0DH or 00001101B) and the value 17 (11H or 00010001B) in B, since 251 = (13 X 18) + 17. Carry and OV will both be cleared.

Bytes: 1

Cycles: 4

Encoding:

1 0 0 0	0 1 0 0

Operation: DIV
$(A)_{15\text{-}8}$
$(B)_{7\text{-}0}$ ← (A)/(B)

DJNZ <byte>,<rel-addr>

Function: Decrement and Jump if Not Zero

Description: DJNZ decrements the location indicated by 1, and branches to the address indicated by the second operand if the resulting value is not zero. An original value of 00H will underflow to 0FFH. No flags are affected. The branch destination would be computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Internal RAM locations 40H, 50H, and 60H contain the values 01H, 70H, and 15H, respectively. The instruction sequence,

```
DJNZ   40H,LABEL__1
DJNZ   50H,LABEL__2
DJNZ   60H,LABEL__3
```

will cause a jump to the instruction at label LABEL__2 with the values 00H, 6FH, and 15H in the three RAM locations. The first jump was *not* taken because the result was zero.

This instruction provides a simple way of executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. The instruction sequence,

```
          MOV     R2,#8
TOGGLE:   CPL     P1.7
          DJNZ    R2,TOGGLE
```

will toggle P1.7 eight times, causing four output pulses to appear at bit 7 of output Port 1. Each pulse will last three machine cycles; two for DJNZ and one to alter the pin.

DJNZ Rn,rel

Bytes: 2

Cycles: 2

Encoding:

1 1 0 1	1 r r r	rel. address

Operation:

```
DJNZ
(PC) ← (PC) + 2
(Rn) ← (Rn) -- 1
IF  (Rn) > 0 or (Rn) < 0
    THEN
          (PC) ← (PC) + rel
```

DJNZ direct,rel

Bytes: 3

Cycles: 2

Encoding:

1 1 0 1	0 1 0 1	direct address	rel. address

Operation:

```
DJNZ
(PC) ← (PC) + 2
(direct) ← (direct) − 1
IF (direct) > 0 or (direct) < 0
   THEN
        (PC) ← (PC) + rel
```

INC <byte>

Function: Increment

Description: INC increments the indicated variable by 1. An original value of 0FFH will overflow to 00H. No flags are affected. Three addressing modes are allowed: register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Register 0 contains 7EH (011111110B). Internal RAM locations 7EH and 7FH contain 0FFH and 40H, respectively. The instruction sequence,

```
INC  @R0
INC  R0
INC  @R0
```

will leave register 0 set to 7FH and internal RAM locations 7EH and 7FH holding (respectively) 00H and 41H.

INC A

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 1 0 0

Operation:

```
INC
(A) ← (A) + 1
```

INC Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	1 r r r

Operation: INC
(Rn) ← (Rn) + 1

INC direct

Bytes: 2

Cycles: 1

Encoding:

0 0 0 0	0 1 0 1	direct address

Operation: INC
(direct) ← (direct) + 1

INC @Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 1 1 i

Operation: INC
((Ri)) ← ((Ri)) + 1

INC DPTR

Function: Increment Data Pointer

Description: Increment the 16-bit data pointer by 1. A 16-bit increment (modulo 2^{16}) is performed; an overflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H will increment the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be incremented.

Example: Registers DPH and DPL contain 12H and 0FEH, respectively. The instruction sequence,

```
INC   DPTR
INC   DPTR
INC   DPTR
```

will change DPH and DPL to 13H and 01H.

Bytes: 1

Cycles: 2

Encoding:

1 0 1 0	0 0 1 1

Operation: INC
(DPTR) ← (DPTR) + 1

JB bit,rel

Function: Jump if Bit set

Description: If the indicated bit is a one, jump to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example: The data present at input port 1 is 11001010B. The Accumulator holds 56 (01010110B). The instruction sequence,

```
JB   P1.2,LABEL1

JB   ACC.2,LABEL2
```

will cause program execution to branch to the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0 0 1 0	0 0 0 0	bit address	rel. address

Operation:

```
JB
(PC) ← (PC) + 3
IF   (bit) = 1
     THEN
          (PC) ← (PC) + rel
```

JBC bit,rel

Function: Jump if Bit is set and Clear bit

Description: If the indicated bit is one, branch to the address indicated; otherwise proceed with the next instruction. *The bit will not be cleared if it is already a zero.* The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected.

Note: When this instruction is used to test an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

Example: The Accumulator holds 56H (01010110B). The instruction sequence,

```
JBC  ACC.3,LABEL1
JBC  ACC.2,LABEL2
```

will cause program execution to continue at the instruction identified by the label LABEL2, with the Accumulator modified to 52H (01010010B).

Bytes: 3

Cycles: 2

Encoding:

0 0 0 1	0 0 0 0	bit address	rel. address

Operation:

```
JBC
(PC) ← (PC) + 3
IF   (bit) = 1
     THEN
          (bit) ← 0
          (PC) ← (PC) + rel
```

JC rel

Function: Jump if Carry is set

Description: If the carry flag is set, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. No flags are affected.

Example: The carry flag is cleared. The instruction sequence,

```
JC    LABEL1
CPL   C
JC    LABEL 2
```

will set the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 0 0	0 0 0 0	rel. address

Operation:

```
JC
(PC) ← (PC) + 2
IF   (C) = 1
     THEN
          (PC) ← (PC) + rel
```

Function: Jump indirect

Description: Add the eight-bit unsigned contents of the Accumulator with the sixteen-bit data pointer, and load the resulting sum to the program counter. This will be the address for subsequent instruction fetches. Sixteen-bit addition is performed (modulo 2^{16}): a carry-out from the low-order eight bits propagates through the higher-order bits. Neither the Accumulator nor the Data Pointer is altered. No flags are affected.

Example: An even number from 0 to 6 is in the Accumulator. The following sequence of instructions will branch to one of four AJMP instructions in a jump table starting at JMP_TBL:

```
        MOV     DPTR,#JMP_TBL
        JMP     @A+DPTR
JMP_TBL: AJMP   LABEL0
        AJMP    LABEL1
        AJMP    LABEL2
        AJMP    LABEL3
```

If the Accumulator equals 04H when starting this sequence, execution will jump to label LABEL2. Remember that AJMP is a two-byte instruction, so the jump instructions start at every other address.

Bytes: 1

Cycles: 2

Encoding:

0 1 1 1	0 0 1 1

Operation: JMP
(PC) ← (A) + (DPTR)

JNB bit,rel

Function: Jump if Bit Not set

Description: If the indicated bit is a zero, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example: The data present at input port 1 is 11001010B. The Accumulator holds 56H (01010110B). The instruction sequence,

```
JNB   P1.3,LABEL1
JNB   ACC.3,LABEL2
```

will cause program execution to continue at the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0 0 1 1	0 0 0 0	bit address	rel. address

Operation:

```
JNB
(PC) ← (PC) + 3
IF  (bit) = 0
    THEN (PC) ← (PC) + rel.
```

JNC rel

Function: Jump if Carry not set

Description: If the carry flag is a zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The carry flag is not modified.

Example: The carry flag is set. The instruction sequence,

```
JNC   LABEL1
CPL   C
JNC   LABEL2
```

will clear the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 0 1	0 0 0 0	rel. address

Operation:

```
JNC
(PC) ← (PC) + 2
IF  (C) = 0
    THEN  (PC) ← (PC) + rel
```

JNZ rel

Function: Jump if Accumulator Not Zero

Description: If any bit of the Accumulator is a one, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally holds 00H. The instruction sequence,

```
JNZ   LABEL1
INC   A
JNZ   LABEL2
```

will set the Accumulator to 01H and continue at label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 1 1	0 0 0 0	rel. address

Operation:

```
JNZ
(PC) ← (PC) + 2
IF    (A) ≠ 0
      THEN   (PC) ← (PC) + rel
```

JZ rel

Function: Jump if Accumulator Zero

Description: If all bits of the Accumulator are zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally contains 01H. The instruction sequence,

```
JZ    LABEL1
DEC A
JZ    LABEL2
```

will change the Accumulator to 00H and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 1 0	0 0 0 0	rel. address

Operation:

```
JZ
(PC) ← (PC) + 2
IF    (A) = 0
      THEN   (PC) ← (PC) + rel
```

LCALL addr16

Function: Long call

Description: LCALL calls a subroutine located at the indicated address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first), incrementing the Stack Pointer by two. The high-order and low-order bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 64K-byte program memory address space. No flags are affected.

Example: Initially the Stack Pointer equals 07H. The label "SUBRTN" is assigned to program memory location 1234H. After executing the instruction,

```
LCALL  SUBRTN
```

at location 0123H, the Stack Pointer will contain 09H, internal RAM locations 08H and 09H will contain 26H and 01H, and the PC will contain 1234H.

Bytes: 3

Cycles: 2

Encoding:

0 0 0 1	0 0 1 0	addr15-addr8	addr7-addr0

Operation:

LCALL
(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← (PC_{7-0})
(SP) ← (SP) + 1
((SP)) ← (PC_{15-8})
(PC) ← $addr_{15-0}$

LJMP addr16

Function: Long Jump

Description: LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64K program memory address space. No flags are affected.

Example: The label "JMPADR" is assigned to the instruction at program memory location 1234H. The instruction,

```
LJMP  JMPADR
```

at location 0123H will load the program counter with 1234H.

Bytes: 3

Cycles: 2

Encoding:

0 0 0 0	0 0 1 0	addr15-addr8	addr7-addr0

Operation:

LJMP
(PC) ← $addr_{15-0}$

Function: Move byte variable

Description: The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

This is by far the most flexible operation. Fifteen combinations of source and destination addressing modes are allowed.

Example: Internal RAM location 30H holds 40H. The value of RAM location 40H is 10H. The data present at input port 1 is 11001010B (0CAH).

```
MOV   R0,#30H   ;R0 <= 30H
MOV   A,@R0     ;A <= 40H
MOV   R1,A      ;R1 <= 40H
MOV   B,@R1     ;B <= 10H
MOV   @R1,P1    ;RAM (40H) <= 0CAH
MOV   P2,P1     ;P2 #0CAH
```

leaves the value 30H in register 0, 40H in both the Accumulator and register 1, 10H in register B, and 0CAH (11001010B) both in RAM location 40H and output on port 2.

MOV A,Rn

Bytes: 1

Cycles: 1

Encoding:

1 1 1 0	1 r r r

Operation: MOV
(A) ← (Rn)

***MOV A,direct**

Bytes: 2

Cycles: 1

Encoding:

1 1 1 0	0 1 0 1	direct address

Operation: MOV
(A) ← (direct)

MOV A,ACC is not a valid instruction.

MOV A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1 1 1 0	0 1 1 i

Operation:
MOV
(A) ← ((Ri))

MOV A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 1	0 1 0 0	immediate data

Operation:
MOV
(A) ← #data

MOV Rn,A

Bytes: 1

Cycles: 1

Encoding:

1 1 1 1	1 r r r

Operation:
MOV
(Rn) ← (A)

MOV Rn,direct

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	1 r r r	direct addr.

Operation:
MOV
(Rn) ← (direct)

MOV Rn,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 1	1 r r r	immediate data

Operation:
MOV
(Rn) ← #data

MOV direct,A

Bytes: 2

Cycles: 1

Encoding:

1 1 1 1	0 1 0 1	direct address

Operation: MOV
(direct) ← (A)

MOV direct,Rn

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	1 r r r	direct address

Operation: MOV
(direct) ← (Rn)

MOV direct,direct

Bytes: 3

Cycles: 2

Encoding:

1 0 0 0	0 1 0 1	dir. addr. (src)	dir. addr. (dest)

Operation: MOV
(direct) ← (direct)

MOV direct,@Ri

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 1 1 i	direct addr.

Operation: MOV
(direct) ← ((Ri))

MOV direct,#data

Bytes: 3

Cycles: 2

Encoding:

0 1 1 1	0 1 0 1	direct address	immediate data

Operation: MOV
(direct) ← #data

MOV @Ri,A

Bytes: 1

Cycles: 1

Encoding:

1 1 1 1	0 1 1 i

Operation: MOV
((Ri)) ← (A)

MOV @Ri,direct

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	0 1 1 i	direct addr.

Operation: MOV
((Ri)) ← (direct)

MOV @Ri,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 1	0 1 1 i	immediate data

Operation: MOV
((RI)) ← #data

MOV <dest-bit>,<src-bit>

Function: Move bit data

Description: The Boolean variable indicated by the second operand is copied into the location specified by the first operand. One of the operands must be the carry flag; the other may be any directly addressable bit. No other register or flag is affected.

Example: The carry flag is originally set. The data present at input Port 3 is 11000101B. The data previously written to output Port 1 is 35H (00110101B).

```
MOV  P1.3,C
MOV  C,P3.3
MOV  P1.2,C
```

will leave the carry cleared and change Port 1 to 39H (00111001B).

MOV C,bit

Bytes: 2

Cycles: 1

Encoding:

1 0 1 0	0 0 1 0	bit address

Operation: MOV
(C) ← (bit)

MOV bit,C

Bytes: 2

Cycles: 2

Encoding:

1 0 0 1	0 0 1 0	bit address

Operation: MOV
(bit) ← (C)

MOV DPTR,#data16

Function: Load Data Pointer with a 16-bit constant

Description: The Data Pointer is loaded with the 16-bit constant indicated. The 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the low-order byte. No flags are affected.

This is the only instruction which moves 16 bits of data at once.

Example: The instruction,

MOV DPTR,#1234H

will load the value 1234H into the Data Pointer: DPH will hold 12H and DPL will hold 34H.

Bytes: 3

Cycles: 2

Encoding:

1 0 0 1	0 0 0 0	immed. data15-8	immed. data7-0

Operation: MOV
(DPTR) ← #data$_{15\text{-}0}$
DPH ☐ DPL ← #data$_{15\text{-}8}$ ☐ #data$_{7\text{-}0}$

MOVC A,@A+<base-reg>

Function: Move Code byte

Description: The MOVC instructions load the Accumulator with a code byte, or constant from program memory. The address of the byte fetched is the sum of the original unsigned eight-bit Accumulator contents and the contents of a sixteen-bit base register, which may be either the Data Pointer or the PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the Accumulator; otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.

Example: A value between 0 and 3 is in the Accumulator. The following instructions will translate the value in the Accumulator to one of four values defined by the DB (define byte) directive.

```
REL_PC: INC   A
        MOVC  A,@A+PC
        RET
        DB    66H
        DB    77H
        DB    88H
        DB    99H
```

If the subroutine is called with the Accumulator equal to 01H, it will return with 77H in the Accumulator. The INC A before the MOVC instruction is needed to "get around" the RET instruction above the table. If several bytes of code separated the MOVC from the table, the corresponding number would be added to the Accumulator instead.

MOVC A,@A+DPTR

Bytes: 1

Cycles: 2

Encoding:

1 0 0 1	0 0 1 1

Operation: MOVC
(A) ← ((A) + (DPTR))

MOVC A,@A + PC

Bytes: 1

Cycles: 2

Encoding:

1 0 0 0	0 0 1 1

Operation: MOVC
(PC) ← (PC) + 1
(A) ← ((A) + (PC))

Function: Move External

Description: The MOVX instructions transfer data between the Accumulator and a byte of external data memory, hence the "X" appended to MOV. There are two types of instructions, differing in whether they provide an eight-bit or sixteen-bit indirect address to the external data RAM.

In the first type, the contents of R0 or R1 in the current register bank provide an eight-bit address multiplexed with data on P0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array. For somewhat larger arrays, any output port pins can be used to output higher-order address bits. These pins would be controlled by an output instruction preceding the MOVX.

In the second type of MOVX instruction, the Data Pointer generates a sixteen-bit address. P2 outputs the high-order eight address bits (the contents of DPH) while P0 multiplexes the low-order eight bits (DPL) with data. The P2 Special Function Register retains its previous contents while the P2 output buffers are emitting the contents of DPH. This form is faster and more efficient when accessing very large data arrays (up to 64K bytes), since no additional instructions are needed to set up the output ports.

It is possible in some situations to mix the two MOVX types. A large RAM array with its high-order address lines driven by P2 can be addressed via the Data Pointer, or with code to output high-order address bits to P2 followed by a MOVX instruction using R0 or R1.

Example: An external 256 byte RAM using multiplexed address/data lines (e.g., an Intel 8155 RAM/I/O/Timer) is connected to the 8051 Port 0. Port 3 provides control lines for the external RAM. Ports 1 and 2 are used for normal I/O. Registers 0 and 1 contain 12H and 34H. Location 34H of the external RAM holds the value 56H. The instruction sequence,

```
MOVX   A,@R1
MOVX   @R0,A
```

copies the value 56H into both the Accumulator and external RAM location 12H.

MOVX A,@Ri

Bytes: 1

Cycles: 2

Encoding:

1 1 1 0	0 0 1 i

Operation: MOVX
(A) ← ((Ri))

MOVX A,@DPTR

Bytes: 1

Cycles: 2

Encoding:

1 1 1 0	0 0 0 0

Operation: MOVX
(A) ← ((DPTR))

MOVX @Ri,A

Bytes: 1

Cycles: 2

Encoding:

1 1 1 1	0 0 1 i

Operation: MOVX
((Ri)) ← (A)

MOVX @DPTR,A

Bytes: 1

Cycles: 2

Encoding:

1 1 1 1	0 0 0 0

Operation: MOVX
(DPTR) ← (A)

NOP

Function: No Operation

Description: Execution continues at the following instruction. Other than the PC, no registers or flags are affected.

Example: It is desired to produce a low-going output pulse on bit 7 of Port 2 lasting exactly 5 cycles. A simple SETB/CLR sequence would generate a one-cycle pulse, so four additional cycles must be inserted. This may be done (assuming no interrupts are enabled) with the instruction sequence,

```
CLR    P2.7
NOP
NOP
NOP
NOP
SETB   P2.7
```

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 0 0 0

Operation: NOP
(PC) ← (PC) + 1

MUL AB

Function: Multiply

Description: MUL AB multiplies the unsigned eight-bit integers in the Accumulator and register B. The low-order byte of the sixteen-bit product is left in the Accumulator, and the high-order byte in B. If the product is greater than 255 (0FFH) the overflow flag is set; otehrwise it is cleared. The carry flag is always cleared.

Example: Originally the Accumulator holds the value 80 (50H). Register B holds the value 160 (0A0H). The instruction,

```
MUL   AB
```

will give the product 12,800 (3200H), so B is changed to 32H (00110010B) and the Accumulator is cleared. The overflow flag is set, carry is cleared.

Bytes: 1

Cycles: 4

Encoding:

1 0 1 0	0 1 0 0

Operation: MUL
$(A)_{7-0} \leftarrow (A) \times (B)$
$(B)_{15-8}$

Function: Logical-OR for byte variables

Description: ORL performs the bitwise logical-OR operation between the indicated variables, storing the results in the destination byte. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: If the Accumulator holds 0C3H (11000011B) and R0 holds 55H (01010101B) then the instruction,

```
ORL   A,R0
```

will leave the Accumulator holding the value 0D7H (11010111B).

When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable computed in the Accumulator at run-time. The instruction,

```
ORL   P1,#00110010B
```

will set bits 5, 4, and 1 of output Port 1.

ORL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 0 0	1 r r r

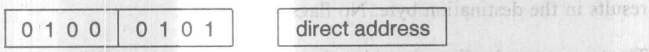
Operation: ORL
(A) ← (A) ∨ (Rn)

ORL A,direct

Bytes: 2

Cycles: 1

Encoding:



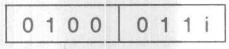
Operation: ORL
(A) ← (A) ∨ (direct)

ORL A,@Ri

Bytes: 1

Cycles: 1

Encoding:



Operation: ORL
(A) ← (A) ∨ ((Ri))

ORL A,#data

Bytes: 2

Cycles: 1

Encoding:



Operation: ORL
(A) ← (A) ∨ #data

ORL direct,A

Bytes: 2

Cycles: 1

Encoding:



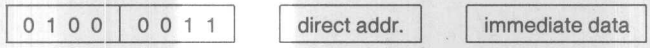
Operation: ORL
(direct) ← (direct) ∨ (A)

ORL direct,#data

Bytes: 3

Cycles: 2

Encoding:



Operation: ORL
(direct) ← (direct) ∨ #data

ORL C,<src-bit>

Function: Logical-OR for bit variables

Description: Set the carry flag if the Boolean value is a logical 1; leave the carry in its current state otherwise . A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

Example: Set the carry flag if and only if P1.0 = 1, ACC. 7 = 1, or OV = 0:

```
MOV   C,P1.0     ;LOAD CARRY WITH INPUT PIN P10
ORL   C,ACC.7    ;OR CARRY WITH THE ACC. BIT 7
ORL   C,/OV      ;OR CARRY WITH THE INVERSE OF OV.
```

ORL C,bit

Bytes: 2

Cycles: 2

Encoding:

0 1 1 1	0 0 1 0	bit address

Operation:

ORL
(C) ← (C) ∨ (bit)

ORL C,/bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	0 0 0 0	bit address

Operation:

ORL
(C) ← (C) ∨ ($\overline{\text{bit}}$)

POP direct

Function: Pop from stack.

Description: The contents of the internal RAM location addressed by the Stack Pointer is read, and the Stack Pointer is decremented by one. The value read is then transferred to the directly addressed byte indicated. No flags are affected.

Example: The Stack Pointer originally contains the value 32H, and internal RAM locations 30H through 32H contain the values 20H, 23H, and 01H, respectively. The instruction sequence,

```
POP   DPH
POP   DPL
```

will leave the Stack Pointer equal to the value 30H and the Data Pointer set to 0123H. At this point the instruction,

```
POP   SP
```

will leave the Stack Pointer set to 20H. Note that in this special case the Stack Pointer was decremented to 2FH before being loaded with the value popped (20H).

Bytes: 2

Cycles: 2

Encoding:

1 1 0 1	0 0 0 0	direct address

Operation:

```
POP
(direct) ← ((SP))
(SP) ← (SP) − 1
```

PUSH direct

Function: Push onto stack

Description: The Stack Pointer is incremented by one. The contents of the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. Otherwise no flags are affected.

Example: On entering an interrupt routine the Stack Pointer contains 09H. The Data Pointer holds the value 0123H. The instruction sequence,

```
PUSH   DPL
PUSH   DPH
```

will leave the Stack Pointer set to 0BH and store 23H and 01H in internal RAM locations 0AH and 0BH, respectively.

Bytes: 2

Cycles: 2

Encoding:

1 1 0 0	0 0 0 0	direct address

Operation:

```
PUSH
(SP) ← (SP) + 1
((SP)) ← (direct)
```

RET

Function: Return from subroutine

Description: RET pops the high- and low-order bytes of the PC successively from the stack, decrementing the Stack Pointer by two. Program execution continues at the resulting address, generally the instruction immediately following an ACALL or LCALL. No flags are affected.

Example: The Stack Pointer originally contains the value 0BH. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

```
RET
```

will leave the Stack Pointer equal to the value 09H. Program execution will continue at location 0123H.

Bytes: 1

Cycles: 2

Encoding:

0 0 1 0	0 0 1 0

Operation:

```
RET
(PC15-8) ← ((SP))
(SP) ← (SP) − 1
(PC7-0) ← ((SP))
(SP) ← (SP) − 1
```

RETI

Function: Return from interrupt

Description: RETI pops the high- and low-order bytes of the PC successively from the stack, and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. The Stack Pointer is left decremented by two. No other registers are affected; the PSW is *not* automatically restored to its pre-interrupt status. Program execution continues at the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower- or same-level interrupt had been pending when the RETI instruction is executed, that one instruction will be executed before the pending interrupt is processed.

Example: The Stack Pointer originally contains the value 0BH. An interrupt was detected during the instruction ending at location 0122H. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

```
RETI
```

will leave the Stack Pointer equal to 09H and return program execution to location 0123H.

Bytes: 1

Cycles: 2

Encoding:

0 0 1 1	0 0 1 0

Operation:

```
RETI
(PC15-8) ← ((SP))
(SP) ← (SP) − 1
(PC7-0) ← ((SP))
(SP) ← (SP) − 1
```

RL A

Function: Rotate Accumulator Left

Description: The eight bits in the Accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

```
RL   A
```

leaves the Accumulator holding the value 8BH (10001011B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0 0 1 0	0 0 1 1

Operation:

```
RL
(An + 1) ← (An)   n = 0 – 6
(A0) ← (A7)
```

RLC A

Function: Rotate Accumulator Left through the Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B), and the carry is zero. The instruction,

```
RLC   A
```

leaves the Accumulator holding the value 8BH (10001010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0 0 1 1	0 0 1 1

Operation:

```
RLC
(An + 1) ← (An)   n = 0 – 6
(A0) ← (C)
(C) ← (A7)
```

RR A

Function: Rotate Accumulator Right

Description: The eight bits in the Accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

RR A

leaves the Accumulator holding the value 0E2H (11100010B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 0 1 1

Operation:

RR
$(An) \leftarrow (A_n + 1) \quad n = 0 - 6$
$(A7) \leftarrow (A0)$

RRC A

Function: Rotate Accumulator Right through Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag; the original value of the carry flag moves into the bit 7 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B), the carry is zero. The instruction,

RRC A

leaves the Accumulator holding the value 62 (01100010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	0 0 1 1

Operation:

RRC
$(An) \leftarrow (An + 1) \quad n = 0 - 6$
$(A7) \leftarrow (C)$
$(C) \leftarrow (A0)$

Function: Set Bit

Description: SETB sets the indicated bit to one. SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

Example: The carry flag is cleared. Output Port 1 has been written with the value 34H (00110100B). The instructions,

```
SETB  C
SETB  P1.0
```

will leave the carry flag set to 1 and change the data output on Port 1 to 35H (00110101B).

SETB C

Bytes: 1

Cycles: 1

Encoding:

1 1 0 1	0 0 1 1

Operation:
SETB
(C) ← 1

SETB bit

Bytes: 2

Cycles: 1

Encoding:

1 1 0 1	0 0 1 0	bit address

Operation:
SETB
(bit) ← 1

Function: Short Jump

Description: Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

Example: The label "RELADR" is assigned to an instruction at program memory location 0123H. The instruction,

```
SJMP   RELADR
```

will assemble into location 0100H. After the instruction is executed, the PC will contain the value 0123H.

(*Note:* Under the above conditions the instruction following SJMP will be at 102H. Therefore, the displacement byte of the instruction will be the relative offset (0123H-0102H) = 21H. Put another way, an SJMP with a displacement of 0FEH would be a one-instruction infinite loop.)

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 0 0 0	rel. address

Operation: SJMP
(PC) ← (PC) + 2
(PC) ← (PC) + rel

SUBB A,<src-byte>

Function: Subtract with borrow

Description: SUBB subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7, and clears C otherwise. (If C was set *before* executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision subtraction, so the carry is subtracted from the Accumulator along with the source operand.) AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

When subtracting signed integers OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C9H (11001001B), register 2 holds 54H (01010100B), and the carry flag is set. The instruction,

SUBB A,R2

will leave the value 74H (01110100B) in the accumulator, with the carry flag and AC cleared but OV set.

Notice that 0C9H minus 54H is 75H. The difference between this and the above result is due to the carry (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision subtraction, it should be explicitly cleared by a CLR C instruction.

SUBB A,Rn

Bytes: 1

Cycles: 1

Encoding:

1 0 0 1	1 r r r

Operation: SUBB

(A) ← (A) − (C) − (Rn)

SUBB A,direct

Bytes: 2

Cycles: 1

Encoding:

1 0 0 1	0 1 0 1	direct address

Operation: SUBB
(A) ← (A) − (C) − (direct)

SUBB A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1 0 0 1	0 1 1 i

Operation: SUBB
(A) ← (A) − (C) − ((Ri))

SUBB A,#data

Bytes: 2

Cycles: 1

Encoding:

1 0 0 1	0 1 0 0	immediate data

Operation: SUBB
(A) ← (A) − (C) − #data

SWAP A

Function: Swap nibbles within the Accumulator

Description: SWAP A interchanges the low- and high-order nibbles (four-bit fields) of the Accumulator (bits 3-0 and bits 7-4). The operation can also be thought of as a four-bit rotate instruction. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

SWAP A

leaves the Accumulator holding the value 5CH (01011100B).

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	0 1 0 0

Operation: SWAP
$(A_{3-0}) \rightleftarrows (A_{7-4})$

Function: Exchange Accumulator with byte variable

Description: XCH loads the Accumulator with the contents of the indicated variable, at the same time writing the original Accumulator contents to the indicated variable. The source/destination operand can use register, direct, or register-indirect addressing.

Example: R0 contains the address 20H. The Accumulator holds the value 3FH (00111111B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

```
XCH   A,@R0
```

will leave RAM location 20H holding the values 3FH (00111111B) and 75H (01110101B) in the accumulator.

XCH A,Rn

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	1 r r r

Operation:
XCH
(A) $\rightleftarrows$ (Rn)

XCH A,direct

Bytes: 2

Cycles: 1

Encoding:

1 1 0 0	0 1 0 1	direct address

Operation:
XCH
(A) $\rightleftarrows$ (direct)

XCH A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	0 1 1 i

Operation:
XCH
(A) $\rightleftarrows$ ((Ri))

XCHD A,@Ri

Function: Exchange Digit

Description: XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (bits 7-4) of each register are not affected. No flags are affected.

Example: R0 contains the address 20H. The Accumulator holds the value 36H (00110110B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

```
XCHD  A,@R0
```

will leave RAM location 20H holding the value 76H (01110110B) and 35H (00110101B) in the Accumulator.

Bytes: 1

Cycles: 1

Encoding:

1 1 0 1	0 1 1 i

Operation:

```
XCHD
(A3-0) ⇄ ((Ri3-0))
```

XRL <dest-byte>,<src-byte>

Function: Logical Exclusive-OR for byte variables

Description: XRL performs the bitwise logical Exclusive-OR operation between the indicated variables, storing the results in the destination. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

(*Note:* When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.)

Example: If the Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) then the instruction,

```
XRL  A,R0
```

will leave the Accumulator holding the value 69H (01101001B).

When the destination is a directly addressed byte, this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte, either a constant contained in the instruction or a variable computed in the Accumulator at run-time. The instruction,

```
XRL  P1,#00110001B
```

will complement bits 5, 4, and 0 of output Port 1.

XRL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 1 0	1 r r r

Operation: XRL
(A) ← (A) ∀ (Rn)

XRL A,direct

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 1 0 1	direct address

Operation: XRL
(A) ← (A) ∀ (direct)

XRL A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 1 1 0	0 1 1 i

Operation: XRL
(A) ← (A) ∀ ((Ri))

XRL A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 1 0 0	immediate data

Operation: XRL
(A) ← (A) ∀ #data

XRL direct,A

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 0 1 0	direct address

Operation: XRL
(direct) ← (direct) ∀ (A)

XRL direct, #data

Bytes: 3

Cycles: 2

Encoding:

0 1 1 0	0 0 1 1	direct address	immediate data

Operation: XRL
(direct) ← (direct) ∀ #data

MCS®-51 Data Sheets, Application Notes, Development Support Tools and Index

10

10

MCS®-51 Data Sh
Application Note
Development Sup
Tools and Index

MCS®-51
8-BIT CONTROL-ORIENTED MICROCOMPUTERS
8031/8051
8031AH/8051AH
8032AH/8052AH
8751H/8751H-8

- **High Performance HMOS Process**
- **Internal Timers/Event Counters**
- **2-Level Interrupt Priority Structure**
- **32 I/O Lines (Four 8-Bit Ports)**
- **64K Program Memory Space**
- **Security Feature Protects EPROM Parts Against Software Piracy**
- **Boolean Processor**
- **Bit-Addressable RAM**
- **Programmable Full Duplex Serial Channel**
- **111 Instructions (64 Single-Cycle)**
- **64K Data Memory Space**

The MCS®-51 products are optimized for control applications. Byte-processing and numerical operations on small data structures are facilitated by a variety of fast addressing modes for accessing the internal RAM. The instruction set provides a convenient menu of 8-bit arithmetic instructions, including multiply and divide instructions. Extensive on-chip support is provided for one-bit variables as a separate data type, allowing direct bit manipulation and testing in control and logic systems that require Boolean processing.

Device	Internal Memory		Timers/ Event Counters	Interrupts
	Program	Data		
8052AH	8K x 8 ROM	256 x 8 RAM	3 x 16-Bit	6
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8051	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8032AH	none	256 x 8 RAM	3 x 16-Bit	6
8031AH	none	128 x 8 RAM	2 x 16-Bit	5
8031	none	128 x 8 RAM	2 x 16-Bit	5
8751H	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5
8751H-8	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5

The 8751H is an EPROM version of the 8051AH; that is, the on-chip Program Memory can be electrically programmed, and can be erased by exposure to ultraviolet light. It is fully compatible with its predecessor, the 8751-8, but incorporates two new features: a Program Memory Security bit that can be used to protect the EPROM against unauthorized read-out, and a programmable baud rate modification bit (SMOD). The 8751H-8 is identical to the 8751H but only operates up to 8 MHz.

October 1987
Order Number: 270048-003

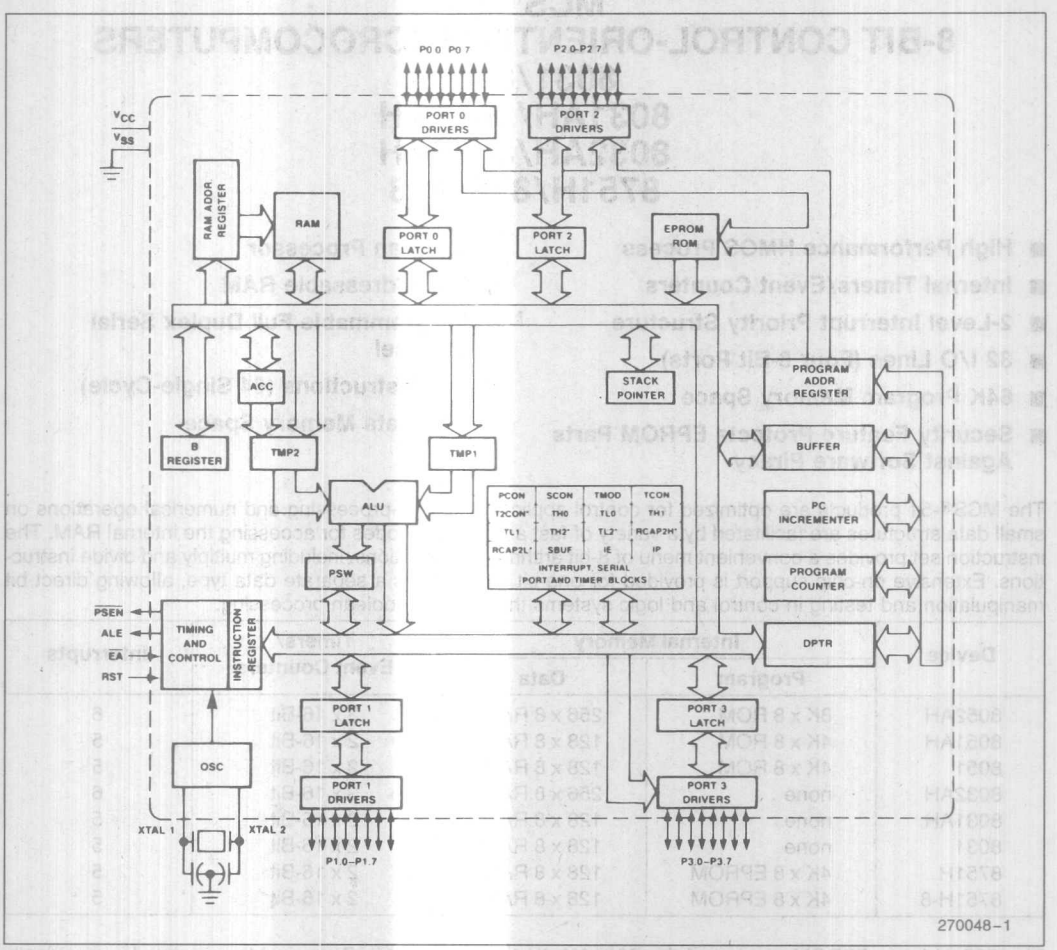


Figure 1. MCS®-51 Block Diagram

PIN DESCRIPTIONS

VCC

Supply voltage.

VSS

Circuit ground.

Port 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s and can source and sink 8 LS TTL inputs.

Port 0 also receives the code bytes during programming of the EPROM parts, and outputs the code bytes during program verification of the ROM and EPROM parts. External pullups are required during program verification.

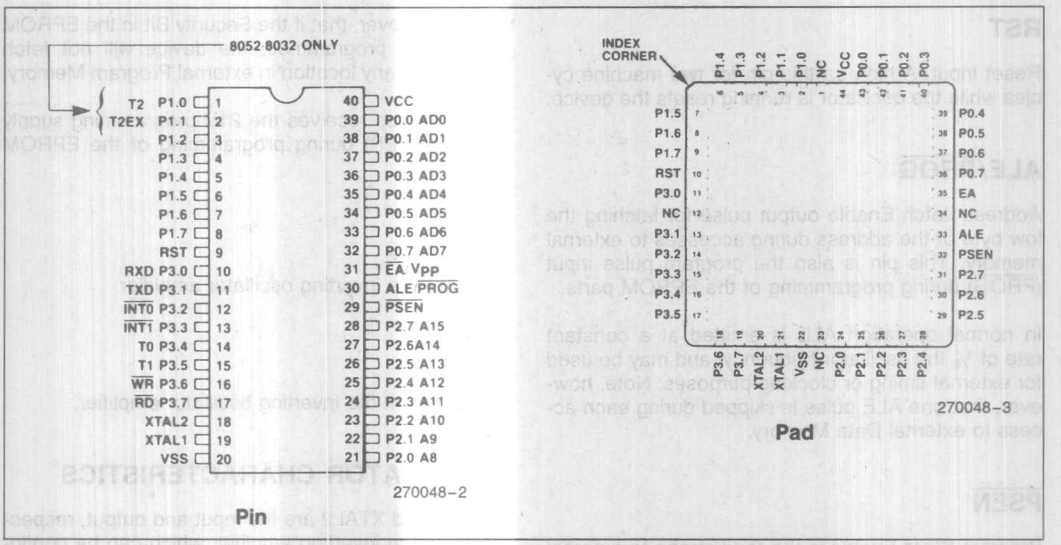


Figure 2. MCS®-51 Connections

Port 1

Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

In the 8032AH and 8052AH, Port 1 pins P1.0 and P1.1 also serve the T2 and T2EX functions, respectively.

Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (IIL on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (IIL on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternative Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	INT0 (external interrupt 0)
P3.3	INT1 (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during programming of the EPROM parts.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

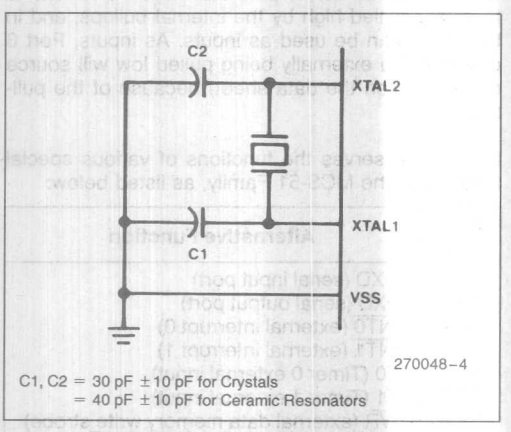
$\overline{PSEN}$

Program Store Enable is the read strobe to external Program Memory.

When the device is executing code from external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external Data Memory.

$\overline{EA}$/VPP

External Access enable $\overline{EA}$ must be strapped to VSS in order to enable any MCS-51 device to fetch code from external Program memory locations 0 to 0FFFH (0 to 1FFFH, in the 8032AH and 8052AH).



C1, C2 = 30 pF ±10 pF for Crystals
= 40 pF ±10 pF for Ceramic Resonators

Figure 3. Oscillator Connections

Note, however, that if the Security Bit in the EPROM devices is programmed, the device will not fetch code from any location in external Program Memory.

This pin also receives the 21V programming supply voltage (VPP) during programming of the EPROM parts.

XTAL1

Input to the inverting oscillator amplifier.

XTAL2

Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

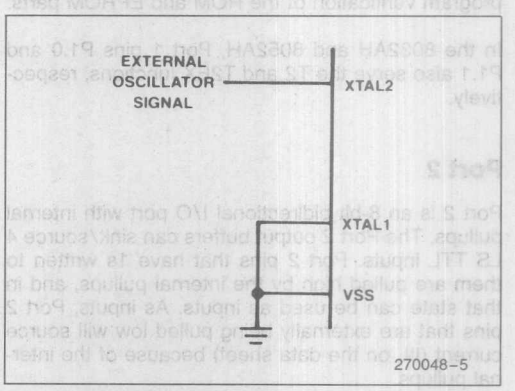


Figure 4. External Drive Configuration

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on $\overline{EA}/V_{PP}$ Pin to V_{SS} ...−0.5V to +21.5V
Voltage on Any Other Pin to V_{SS} ....−0.5V to +7V
Power Dissipation.....1.5W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C; V_{CC} = 5V ±10%; V_{SS} = 0V

Symbol	Parameter		Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (Except $\overline{EA}$ Pin of 8751H & 8751H-8)		−0.5	0.8	V	
V_{IL1}	Input Low Voltage to $\overline{EA}$ Pin of 8751H & 8751H-8		0	0.7	V	
V_{IH}	Input High Voltage (Except XTAL2, RST)		2.0	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage to XTAL2, RST		2.5	V_{CC} + 0.5	V	XTAL1 = V_{SS}
V_{OL}	Output Low Voltage (Ports 1, 2, 3)*			0.45	V	I_{OL} = 1.6 mA
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$)*					
		8751H, 8751H-8		0.60 0.45	V V	I_{OL} = 3.2 mA I_{OL} = 2.4 mA
		All Others		0.45	V	I_{OL} = 3.2 mA
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, $\overline{PSEN}$)		2.4		V	I_{OH} = −80 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)		2.4		V	I_{OH} = −400 μA
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3, RST) 8032AH, 8052AH All Others			−800 −500	μA μA	V_{IN} = 0.45V V_{IN} = 0.45V
I_{IL1}	Logical 0 Input Current to $\overline{EA}$ Pin of 8751H & 8751H-8 Only			−15	mA	
I_{IL2}	Logical 0 Input Current (XTAL2)			−3.2	mA	V_{IN} = 0.45V
I_{LI}	Input Leakage Current (Port 0) 8751H & 8751H-8 All Others			±100 ±10	μA μA	0.45 ≤ V_{IN} ≤ V_{CC} 0.45 ≤ V_{IN} ≤ V_{CC}
I_{IH}	Logical 1 Input Current to $\overline{EA}$ Pin of 8751H & 8751H-8			500	μA	
I_{IH1}	Input Current to RST to Activate Reset			500	μA	V_{IN} < (V_{CC} − 1.5V)
I_{CC}	Power Supply Current: 8031/8051 8031AH/8051AH 8032AH/8052AH 8751H/8751H-8			 160 125 175 250	 mA mA mA mA	 All Outputs Disconnected; $\overline{EA}$ = V_{CC}
C_{IO}	Pin Capacitance			10	pF	Test freq = 1 MHz

***NOTE:**
Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

A.C. CHARACTERISTICS

T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V;
Load Capacitance for Port 0, ALE, and PSEN = 100 pF;
Load Capacitance for All Other Outputs = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL−40		ns
TLLAX	Address Hold after ALE Low	48		TCLCL−35		ns
TLLIV	ALE Low to Valid Instr In					
	8751H		183		4TCLCL−150	ns
	All Others		233		4TCLCL−100	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	58		TCLCL−25		ns
TPLPH	$\overline{PSEN}$ Pulse Width					
	8751H	190		3TCLCL−60		ns
	All Others	215		3TCLCL−35		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr In					
	8751H		100		3TCLCL−150	ns
	All Others		125		3TCLCL−125	ns
TPXIX	Input Instr Hold after $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float after $\overline{PSEN}$		63		TCLCL−20	ns
TPXAV	$\overline{PSEN}$ to Address Valid	75		TCLCL−8		ns
TAVIV	Address to Valid Instr In					
	8751H		267		5TCLCL−150	ns
	All Others		302		5TCLCL−115	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		20		20	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL−100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL−100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		252		5TCLCL−165	ns
TRHDX	Data Hold after $\overline{RD}$	0		0		ns
TRHDZ	Data Float after $\overline{RD}$		97		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL−150	ns
TAVDV	Address to Valid Data In		585		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition					
	8751H	13		TCLCL−70		ns
	All Others	23		TCLCL−60		ns
TQVWH	Data Valid to $\overline{WR}$ High	433		7TCLCL−150		ns
TWHQX	Data Hold after $\overline{WR}$	33		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		20		20	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High					
	8751H	33	133	TCLCL−50	TCLCL+50	ns
	All Others	43	123	TCLCL−40	TCLCL+40	ns

NOTE:
*This table does not include the 8751-8 A.C. characteristics (see next page).

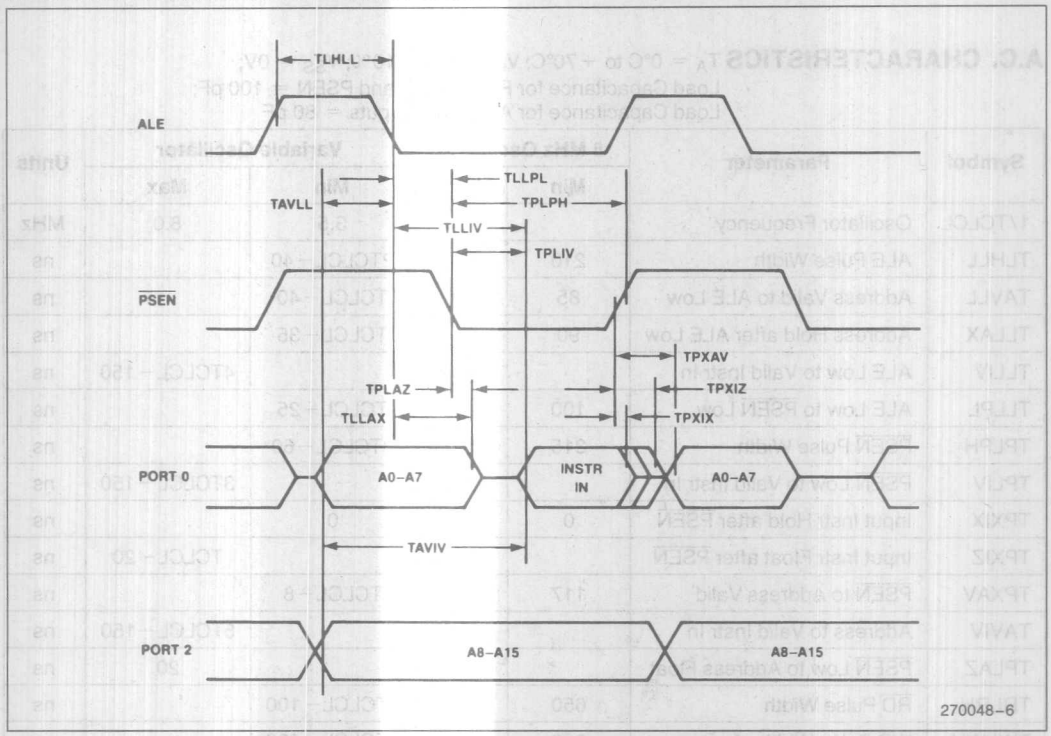
This Table is only for the 8751H-8

A.C. CHARACTERISTICS

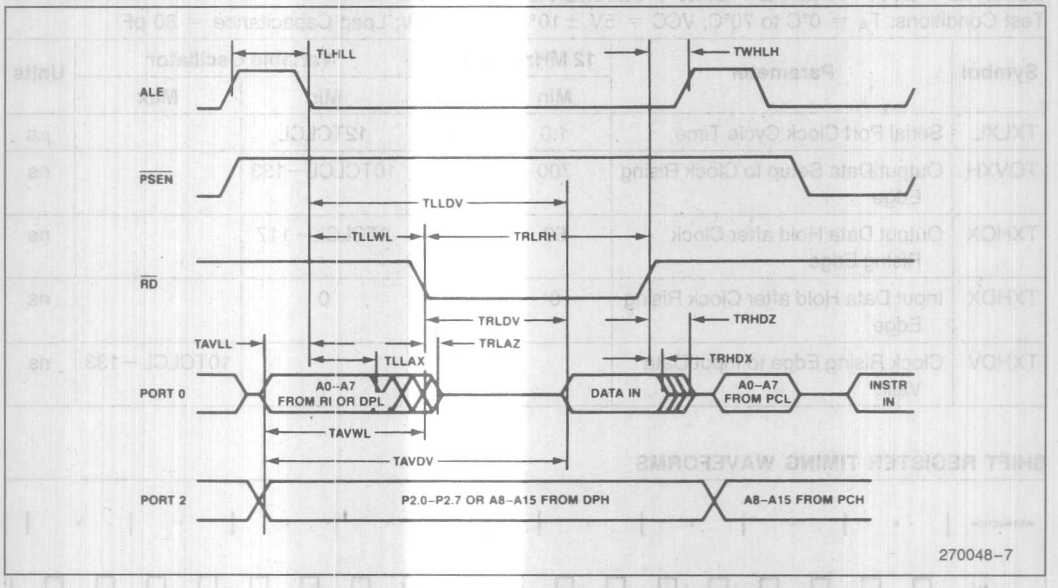
$T_A = 0°C$ to $+70°C$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$;
Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF;
Load Capacitance for All Other Outputs = 80 pF

Symbol	Parameter	8 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	8.0	MHz
TLHLL	ALE Pulse Width	210		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	85		TCLCL−40		ns
TLLAX	Address Hold after ALE Low	90		TCLCL−35		ns
TLLIV	ALE Low to Valid Instr In		350		4TCLCL−150	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	100		TCLCL−25		ns
TPLPH	$\overline{PSEN}$ Pulse Width	315		3TCLCL−60		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr In		225		3TCLCL−150	ns
TPXIX	Input Instr Hold after $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float after $\overline{PSEN}$		105		TCLCL−20	ns
TPXAV	$\overline{PSEN}$ to Address Valid	117		TCLCL−8		ns
TAVIV	Address to Valid Instr In		475		5TCLCL−150	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		20		20	ns
TRLRH	$\overline{RD}$ Pulse Width	650		6TCLCL−100		ns
TWLWH	$\overline{WR}$ Pulse Width	650		6TCLCL−100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		460		5TCLCL−165	ns
TRHDX	Data Hold after $\overline{RD}$	0		0		ns
TRHDZ	Data Float after $\overline{RD}$		180		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		850		8TCLCL−150	ns
TAVDV	Address to Valid Data In		960		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	325	425	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	370		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition	55		TCLCL−70		ns
TQVWH	Data Valid to $\overline{WR}$ High	725		7TCLCL−150		ns
TWHQX	Data Hold after $\overline{WR}$	75		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		20		20	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	75	175	TCLCL−50	TCLCL+50	ns

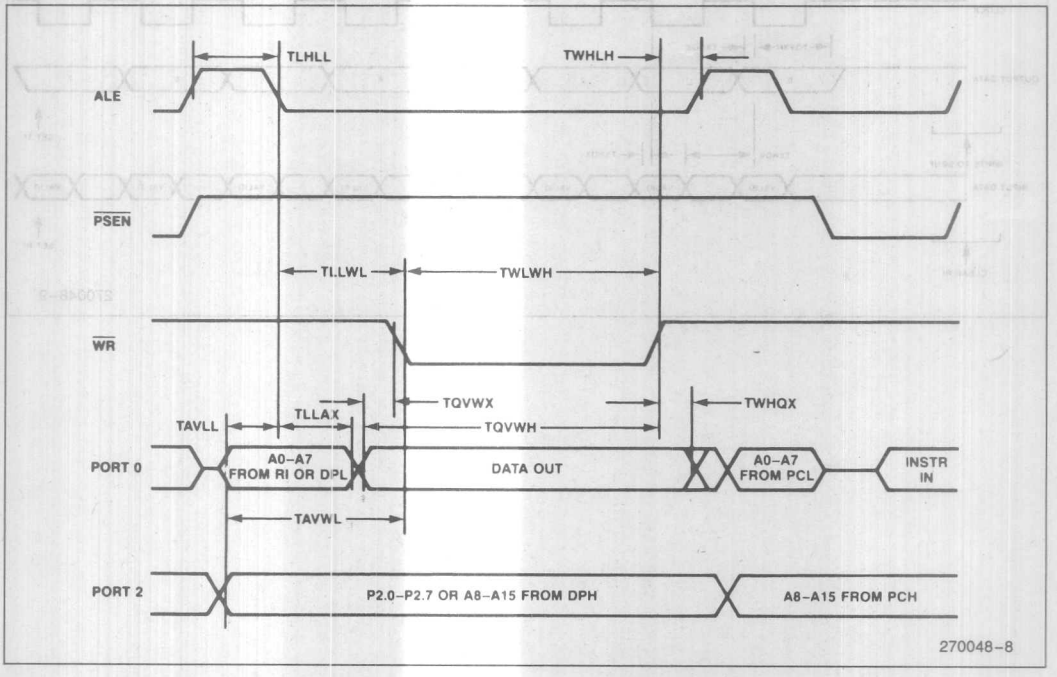
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE

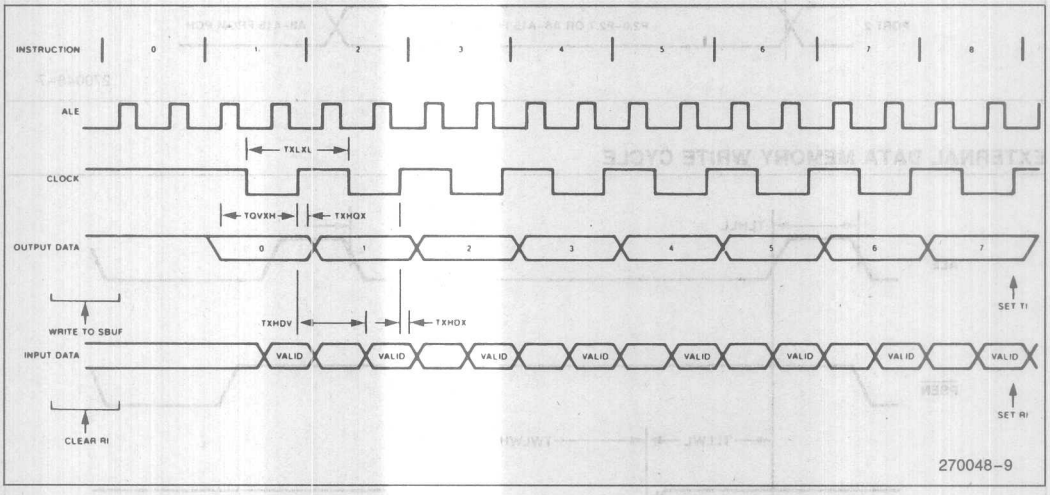


SERIAL PORT TIMING—SHIFT REGISTER MODE

Test Conditions: T_A = 0°C to 70°C; VCC = 5V ±10%; VSS = 0V; Load Capacitance = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL−133		ns
TXHQX	Output Data Hold after Clock Rising Edge	50		2TCLCL−117		ns
TXHDX	Input Data Hold after Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL−133	ns

SHIFT REGISTER TIMING WAVEFORMS

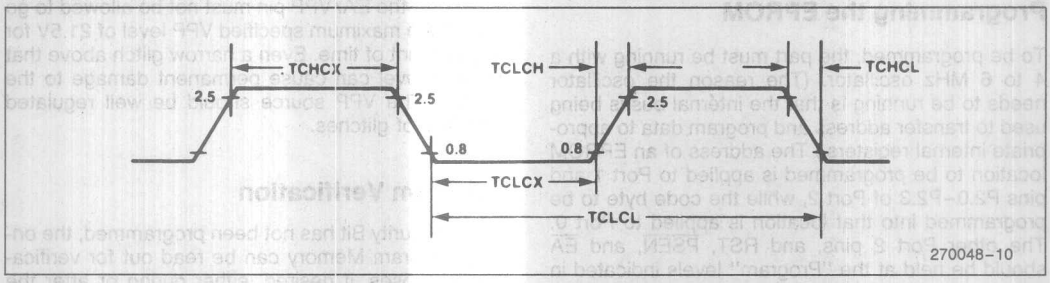


270048-9

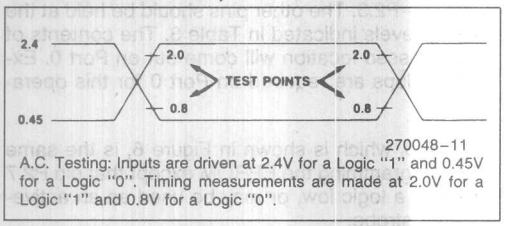
EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency (except 8751H-8)	3.5	12	MHz
	8751H-8	3.5	8	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

EXTERNAL CLOCK DRIVE WAVEFORM



A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

EPROM CHARACTERISTICS

Table 3. EPROM Programming Modes

Mode	RST	$\overline{PSEN}$	ALE	$\overline{EA}$	P2.7	P2.6	P2.5	P2.4
Program	1	0	0*	VPP	1	0	X	X
Inhibit	1	0	1	X	1	0	X	X
Verify	1	0	1	1	0	0	X	X
Security Set	1	0	0*	VPP	1	1	X	X

NOTE:
"1" = logic high for that pin
"0" = logic low for that pin
"X" = "don't care"
"VPP" = +21V ±0.5V
*ALE is pulsed low for 50 ms.

Programming the EPROM

To be programmed, the part must be running with a 4 to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate internal registers.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0–P2.3 of Port 2, while the code byte to be programmed into that location is applied to Port 0. The other Port 2 pins, and RST, $\overline{PSEN}$, and $\overline{EA}$ should be held at the "Program" levels indicated in Table 3. ALE is **pulsed** low for 50 ms to program the code byte into the addressed EPROM location. The setup is shown in Figure 5.

Normally $\overline{EA}$ is held at a logic high until just before ALE is to be pulsed. Then $\overline{EA}$ is raised to +21V, ALE is pulsed, and then $\overline{EA}$ is returned to a logic high. Waveforms and detailed timing specifications are shown in later sections of this data sheet.

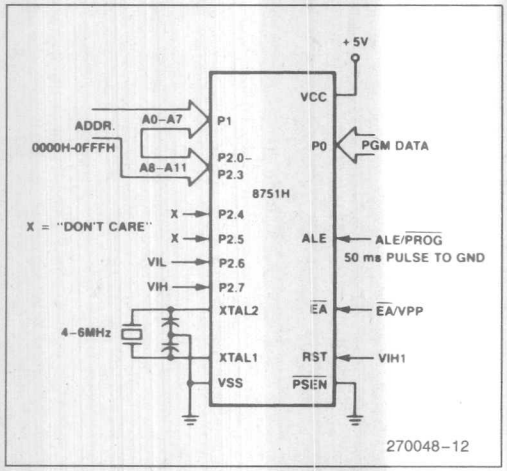


Figure 5. Programming Configuration

Note that the $\overline{EA}$/VPP pin must not be allowed to go above the maximum specified VPP level of 21.5V for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The VPP source should be well regulated and free of glitches.

Program Verification

If the Security Bit has not been programmed, the on-chip Program Memory can be read out for verification purposes, if desired, either during or after the programming operation. The address of the Program Memory location to be read is applied to Port 1 and pins P2.0–P2.3. The other pins should be held at the "Verify" levels indicated in Table 3. The contents of the addressed location will come out on Port 0. External pullups are required on Port 0 for this operation.

The setup, which is shown in Figure 6, is the same as for programming the EPROM except that pin P2.7 is held at a logic low, or may be used as an active-low read strobe.

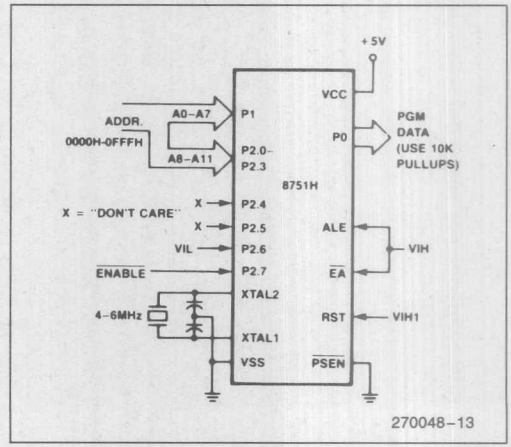


Figure 6. Program Verification

EPROM Security

The security feature consists of a "locking" bit which when programmed denies electrical access by any external means to the on-chip Program Memory. The bit is programmed as shown in Figure 7. The setup and procedure are the same as for normal EPROM programming, except that P2.6 is held at a logic high. Port 0, Port 1, and pins P2.0–P2.3 may be in any state. The other pins should be held at the "Security" levels indicated in Table 3.

Once the Security Bit has been programmed, it can be cleared only by full erasure of the Program Memory. While it is programmed, the internal Program Memory can not be read out, the device can not be further programmed, and it **can not execute out of external program memory**. Erasing the EPROM, thus clearing the Security Bit, restores the device's full functionality. It can then be reprogrammed.

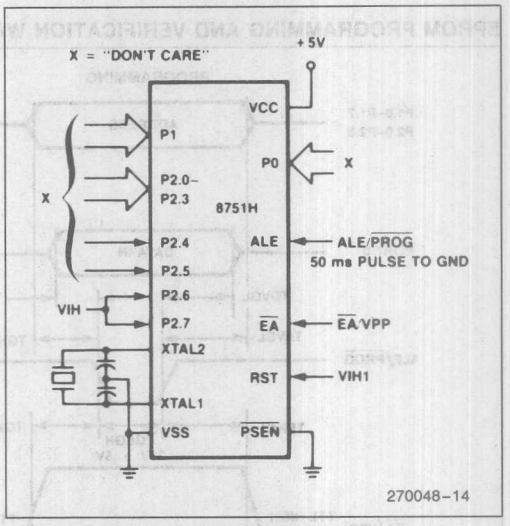


Figure 7. Programming the Security Bit

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at least 15 W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000 μW/cm^2 rating for 20 to 30 minutes, at a distance of about 1 inch, should be sufficient.

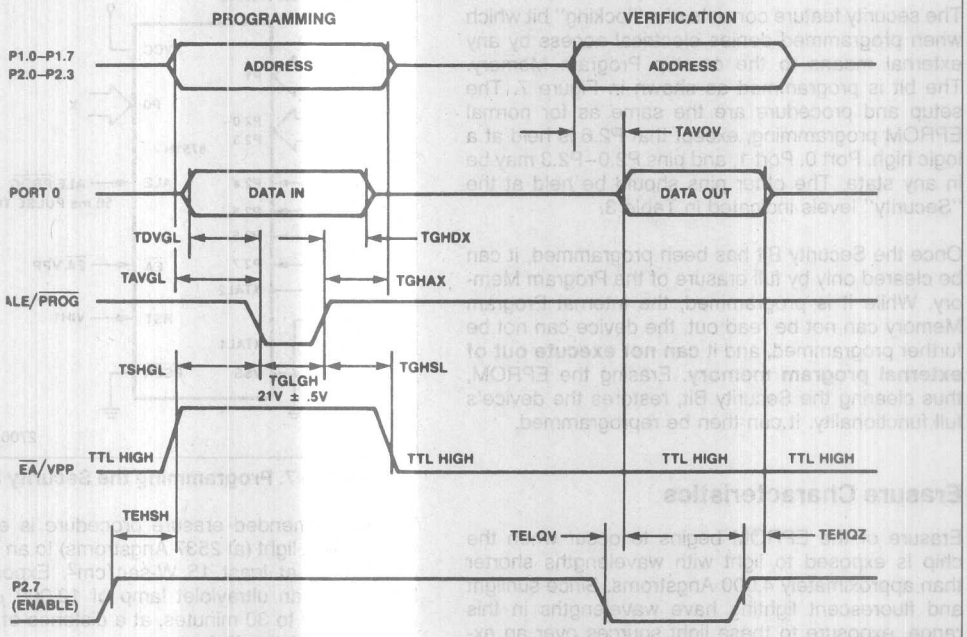
Erasure leaves the array in an all 1s state.

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_A = 21°C to 27°C; VCC = 5V ±10%; VSS = 0V

Symbol	Parameter	Min	Max	Units
VPP	Programming Supply Voltage	20.5	21.5	V
IPP	Programming Supply Current		30	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHAX	Address Hold after $\overline{\text{PROG}}$	48TCLCL		
TDVGL	Data Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHDX	Data Hold after $\overline{\text{PROG}}$	48TCLCL		
TEHSH	P2.7 ($\overline{\text{ENABLE}}$) High to VPP	48TCLCL		
TSHGL	VPP Setup to $\overline{\text{PROG}}$ Low	10		µs
TGHSL	VPP Hold after $\overline{\text{PROG}}$	10		µs
TGLGH	$\overline{\text{PROG}}$ Width	45	55	ms
TAVQV	Address to Data Valid		48TCLCL	
TELQV	$\overline{\text{ENABLE}}$ Low to Data Valid		48TCLCL	
TEHQZ	Data Float after $\overline{\text{ENABLE}}$	0	48TCLCL	

EPROM PROGRAMMING AND VERIFICATION WAVEFORMS



For programming conditions see Figure 5.

For verification conditions see Figure 6.

8051AHP
MCS®-51 FAMILY 8-BIT CONTROL-ORIENTED MICROCONTROLLER WITH PROTECTED ROM

- **High Performance HMOS Process**
- **Internal Timers/Event Counters**
- **2-Level Interrupt Priority Structure**
- **32 I/O Lines (Four 8-Bit Ports)**
- **4K Program Memory Space**
- **Protection Feature Protects ROM Parts Against Software Piracy**
- **Boolean Processor**
- **Bit-Addressable RAM**
- **Programmable Full Duplex Serial Channel**
- **111 Instructions (64 Single-Cycle)**
- **4K Data Memory Space***
  *Expandable to 64K
- **Available in 40 Pin Plastic and CERDIP Packages**
  (See Packaging Outlines and Dimensions Order #231369)

The MCS®-51 products are optimized for control applications. Byte-processing and numerical operations on small data structures are facilitated by a variety of fast addressing modes for accessing the internal RAM. The instruction set provides a convenient menu of 8-bit arithmetic instructions, including multiply and divide instructions. Extensive on-chip support is provided for one-bit variables as a separate data type, allowing direct bit manipulation and testing in control and logic systems that require Boolean processing.

MCS-51 HMOS Family Device	Internal Memory		Timers/ Event Counters	Interrupts
	Program	Data		
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8051AHP	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5

The 8051AHP is identical to the 8051AH with the exception of the Protection Feature. To incorporate this Protection Feature, program verification has been disabled and external memory accesses have been limited to 4K.

September 1987
Order Number: 270279-002

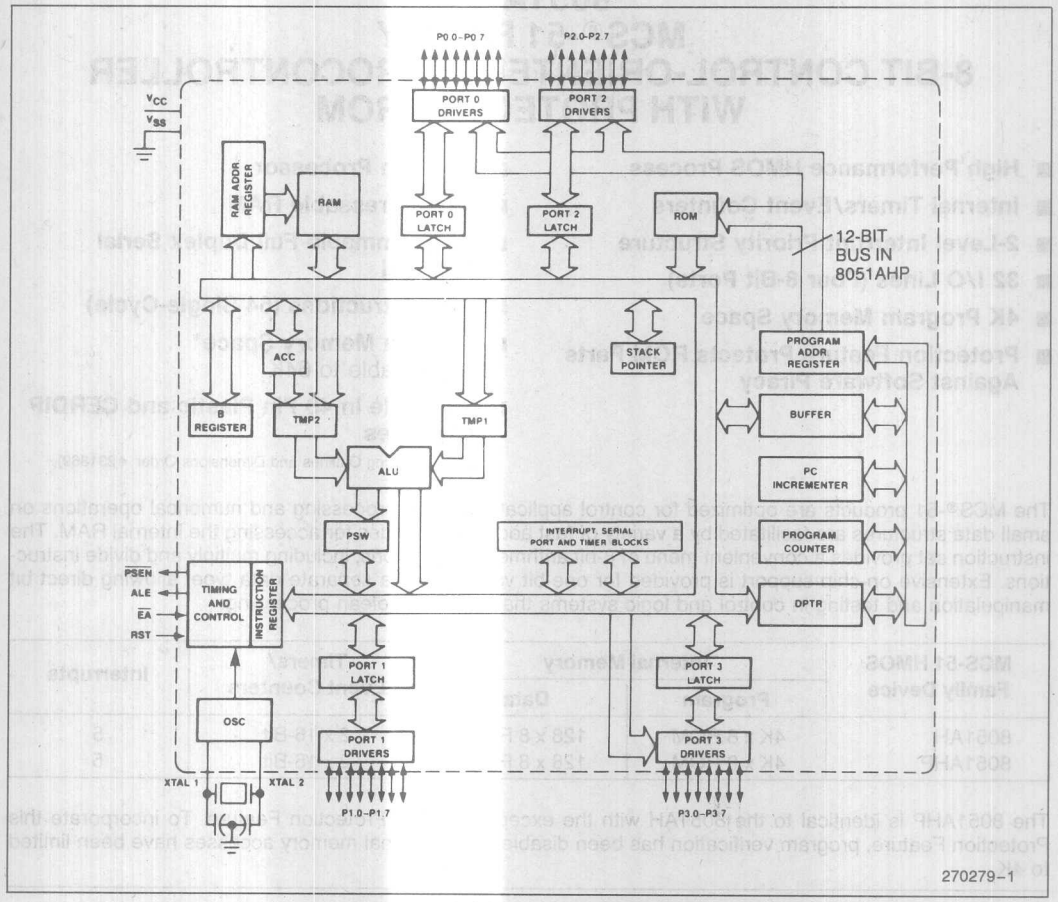


Figure 1. MCS®-51 Block Diagram

PIN DESCRIPTIONS

V_{CC}

Supply voltage.

V_{SS}

Circuit ground.

Port 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs.

Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s and can source and sink 8 LS TTL inputs.

Port 0 also receives the code bytes during programming of the EPROM parts, and outputs the code bytes during program verification of the ROM and EPROM parts. External pullups are required during program verification.

Port 1

Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink source 4 LS TTL inputs. Port 1 pins that have 1s written to

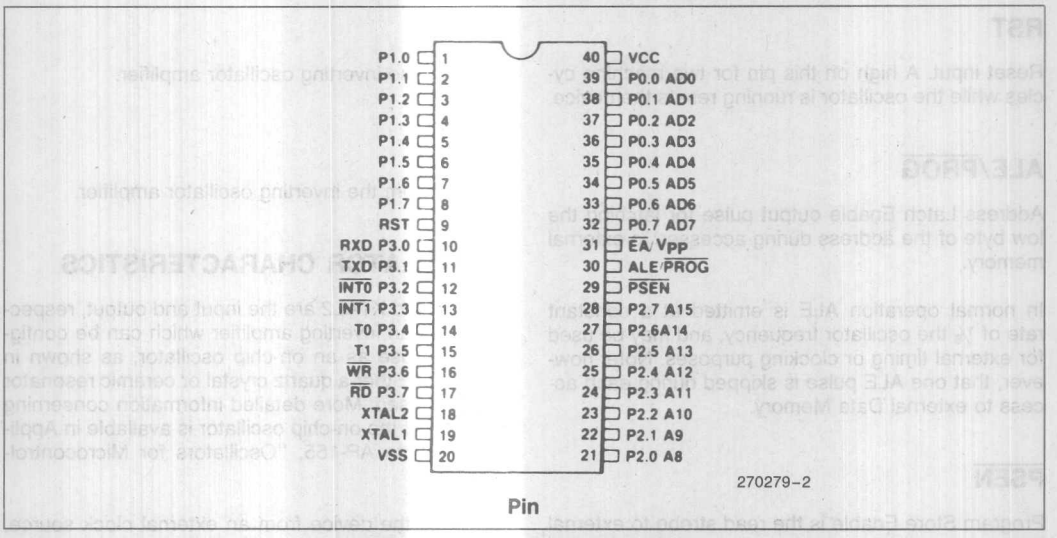


Figure 2. MCS®-51 Connections

them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. Bits P2.4 through P2.7 are forced to 0, effectively limiting external Data and Code space to 4K each in the 8051AHP during external accesses*. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

*Protection feature

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternative Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

$\overline{PSEN}$

Program Store Enable is the read strobe to external Program Memory.

When the device is executing code from external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external Data Memory.

$\overline{EA}/V_{PP}$

External Access enable $\overline{EA}$ should be strapped to V_{CC} for internal program executions. $\overline{EA}$ must be strapped to V_{SS} in order to enable any MCS-51 device to fetch code from external Program memory locations 0 to 0FFFH.

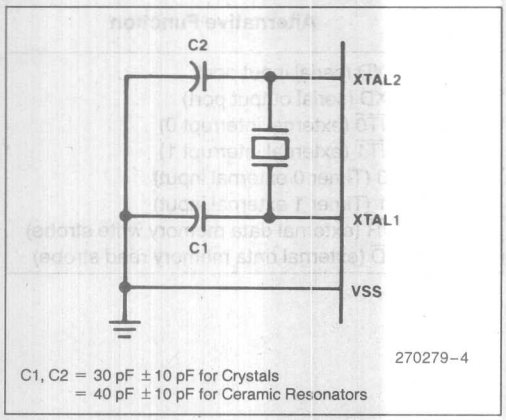


Figure 3. Oscillator Connections

XTAL1

Input to the inverting oscillator amplifier.

XTAL2

Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

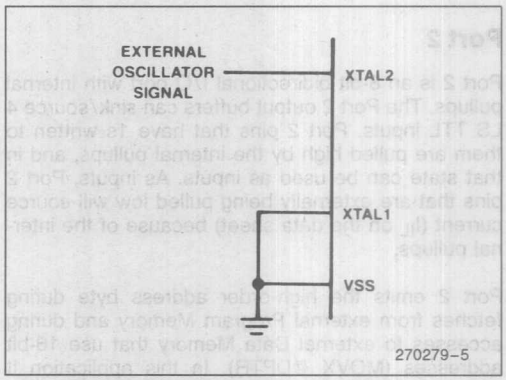


Figure 4. External Drive Configuration

DESIGN CONSIDERATION

The 8051AHP cannot access external Program or Data memory above 4K. This means that the following instructions that use the Data Pointer only read/write data at address locations below 4K:

```
MOVX A, @DPTR
MOVX @DPTR, A
```

When the Data Pointer contains an address above the 4K limit, those locations will not be accessed.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on $\overline{EA}/V_{PP}$ Pin to V_{SS} . . . −0.5V to +21.5V
Voltage on Any Other Pin to V_{SS} . . . . −0.5V to +7V
Power Dissipation 1.5W

Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage (Except XTAL2, RST)	2.0	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage to XTAL2, RST	2.5	V_{CC} + 0.5	V	XTAL1 = V_{SS}
V_{OL}	Output Low Voltage (Ports 1, 2, 3)*		0.45	V	I_{OL} = 1.6 mA
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$)*		0.45	V	I_{OL} = 3.2 mA
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, $\overline{PSEN}$)	2.4		V	I_{OH} = −80 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4		V	I_{OH} = −400 μA
I_{IL}	Logical 0 Input Current		−500	μA	V_{IN} = 0.45V
I_{IL2}	Logical 0 Input Current (XTAL2)		−3.2	mA	V_{IN} = 0.45V
I_{LI}	Input Leakage Current (Port 0)		±10	μA	0.45 ≤ V_{IN} ≤ V_{CC}
I_{IH}	Input Current to RST to Activate Reset		500	μA	V_{IN} < (V_{CC} − 1.5V)
I_{CC}	Power Supply Current		125	mA	All Outputs Disconnected; $\overline{EA}$ = VCC
CIO	Pin Capacitance		10	pF	Test freq = 1 MHz

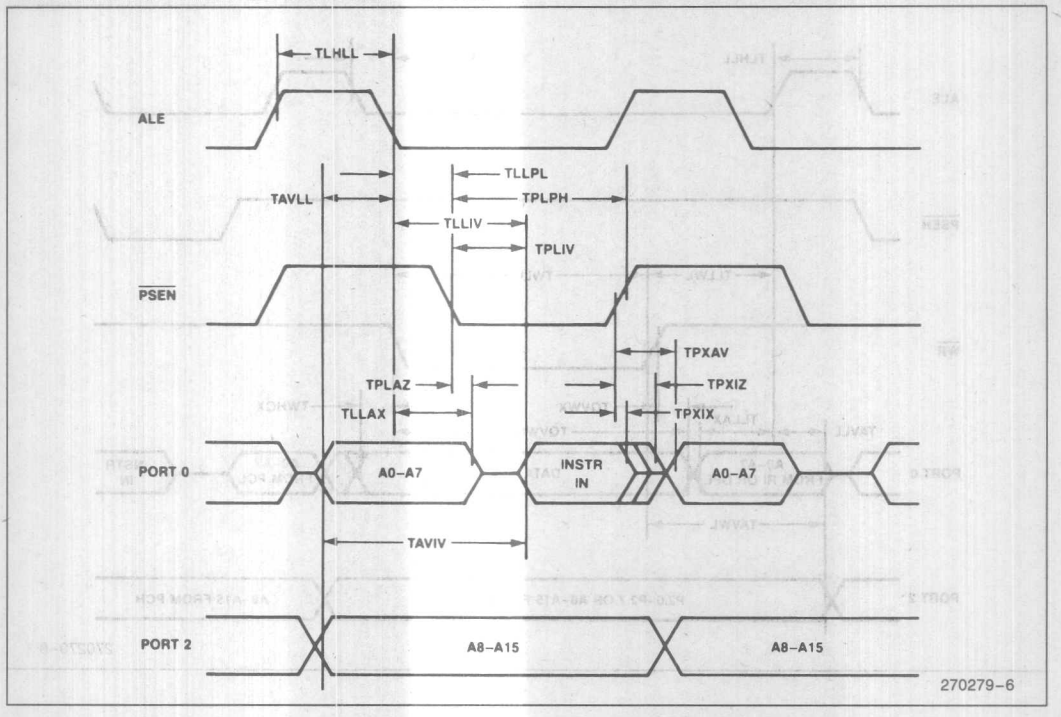
***NOTE:**
Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

A.C. CHARACTERISTICS

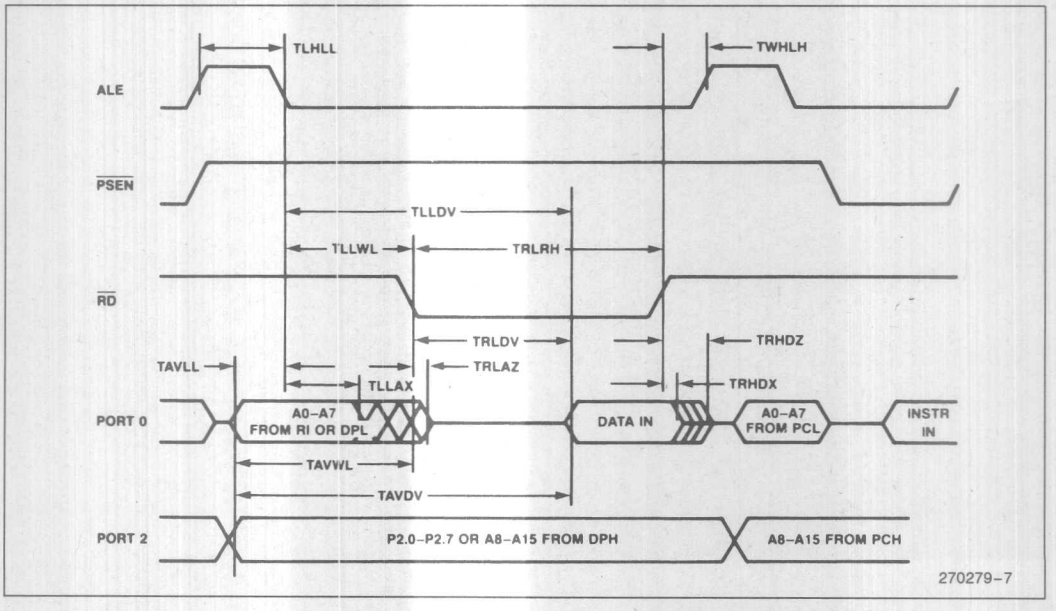
T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V;
Load Capacitance for Port 0, ALE, and PSEN = 100 pF;
Load Capacitance for All Other Outputs = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL−40		ns
TLLAX	Address Hold after ALE Low	48		TCLCL−35		ns
TLLIV	ALE Low to Valid Instr In		233		4TCLCL−100	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	58		TCLCL−25		ns
TPLPH	$\overline{PSEN}$ Pulse Width	215		3TCLCL−35		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr In		125		3TCLCL−125	ns
TPXIX	Input Instr Hold after $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float after $\overline{PSEN}$		63		TCLCL−20	ns
TPXAV	$\overline{PSEN}$ to Address Valid	75		TCLCL−8		ns
TAVIV	Address to Valid Instr In		302		5TCLCL−115	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		20		20	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL−100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL−100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		252		5TCLCL−165	ns
TRHDX	Data Hold after $\overline{RD}$	0		0		ns
TRHDZ	Data Float after $\overline{RD}$		97		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL−150	ns
TAVDV	Address to Valid Data In		585		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition	23		TCLCL−60		ns
TQVWH	Data Valid to $\overline{WR}$ High	433		7TCLCL−150		ns
TWHQX	Data Hold after $\overline{WR}$	33		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		20		20	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	TCLCL−40	TCLCL+40	ns

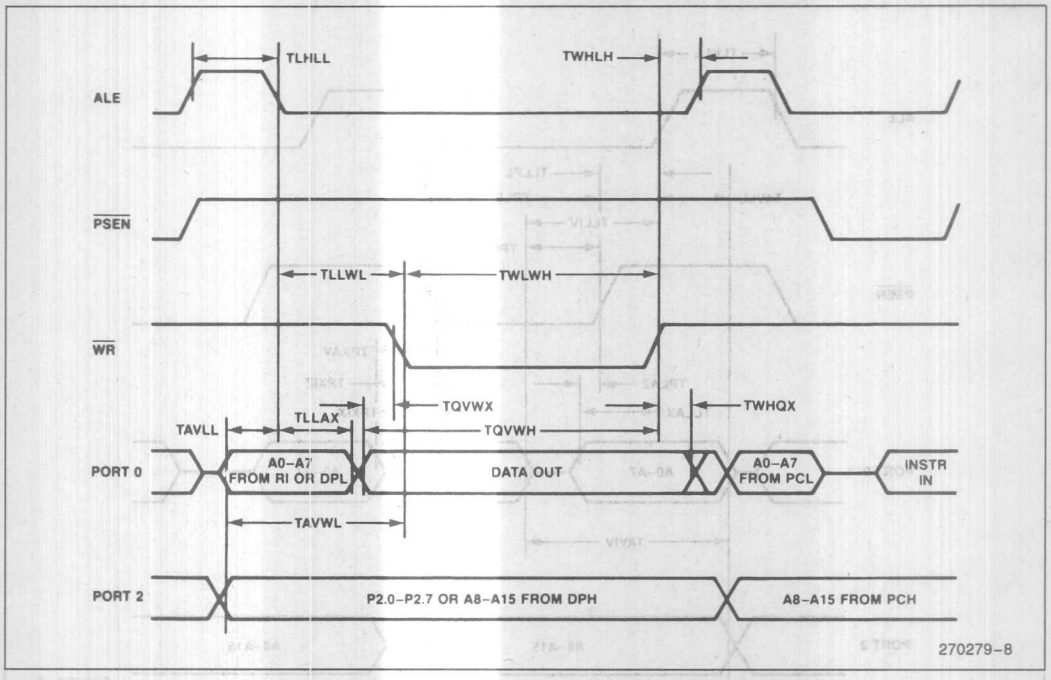
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE

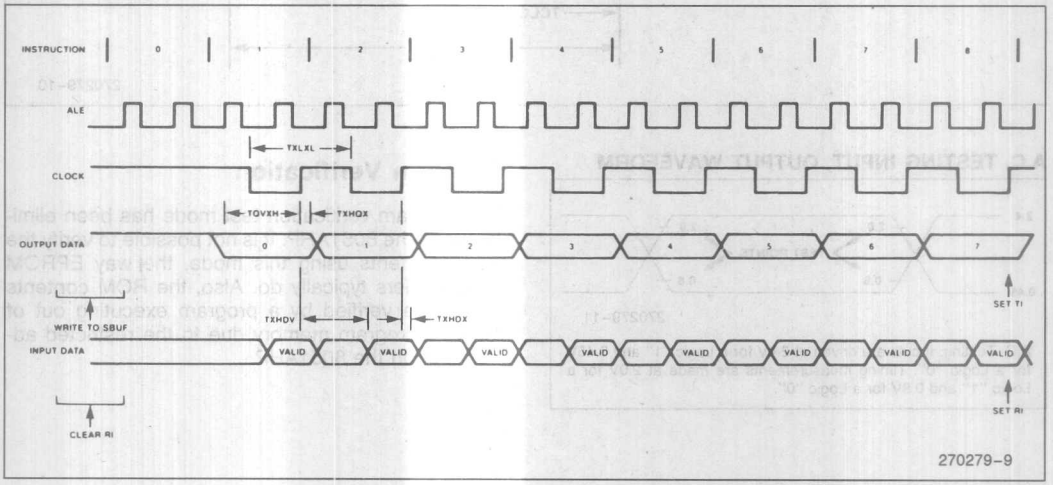


SERIAL PORT TIMING—SHIFT REGISTER MODE

Test Conditions: T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL−133		ns
TXHQX	Output Data Hold after Clock Rising Edge	50		2TCLCL−117		ns
TXHDX	Input Data Hold after Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL−133	ns

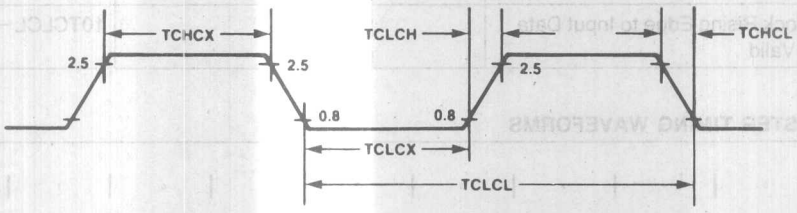
SHIFT REGISTER TIMING WAVEFORMS



EXTERNAL CLOCK DRIVE

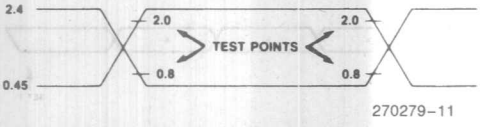
Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

EXTERNAL CLOCK DRIVE WAVEFORM



270279-10

A.C. TESTING INPUT, OUTPUT WAVEFORM



270279-11

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

Program Verification

The program verification test mode has been eliminated on the 8051AHP. It is not possible to verify the ROM contents using this mode, the way EPROM programmers typically do. Also, the ROM contents cannot be verified by a program executing out of external program memory due to the restricted addressing on the 8051AHP.

8031AH/8051AH
8032AH/8052AH
8751H/8751H-8

EXPRESS

■ **Extended Temperature Range** ■ **Burn-In**

The Intel EXPRESS system offers enhancements to the operational specifications of the MCS®-51 family of microcontrollers. These EXPRESS products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

The EXPRESS program includes the commercial standard temperature range with burn-in, and an extended temperature range with or without burn-in.

With the commercial standard temperature range operational characteristics are guaranteed over the temperature range of 0°C to 70°C. With the extended temperature range option, operational characteristics are guaranteed over the range of −40°C to +85°C.

The optional burn-in is dynamic, for a minimum time of 160 hours at 125°C with V_{CC} = 5.5V ±0.25V, following guidelines in MIL-STD-883, Method 1015.

Package types and EXPRESS versions are identified by a one- or two-letter prefix to the part number. The prefixes are listed in Table 1.

For the extended temperature range option, this data sheet specifies the parameters which deviate from their commercial temperature range limits. The commercial temperature range data sheets are applicable for all parameters not listed here.

Electrical Deviations from Commercial Specifications for Extended Temperature Range

D.C. and A.C. parameters not included here are the same as in the commercial temperature range data sheets.

D.C. CHARACTERISTICS T_A = −40°C to +85°C; V_{CC} = 5V ±10%; V_{SS} = 0V

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.75	V	
V_{IH}	Input High Voltage (Except XTAL2, RST)	2.1	V_{CC} + 0.5	V	
I_{CC}	Power Supply Current: 8051AH, 8031AH 8052AH, 8032AH 8751H, 8751H-8		 135 175 265	 mA mA mA	 All Outputs Disconnected; $\overline{EA}$ = V_{CC}
I_{IL2}	Logic 0 Input Current (XTAL2)		−4.0	mA	V_{in} = 0.45V

October 1987
Order Number: 270007-002

Table 1. Prefix Identification

Prefix	Package Type	Temperature Range	Burn-In
P	plastic	commercial	no
D	cerdip	commercial	no
C	ceramic	commercial	no
N	PLCC	commercial	no
R	LCC	commercial	no
TP	plastic	extended	no
TD	cerdip	extended	no
TC	ceramic	extended	no
QP	plastic	commercial	yes
QD	cerdip	commercial	yes
QC	ceramic	commercial	yes
LP	plastic	extended	yes
LD	cerdip	extended	yes
LC	ceramic	extended	yes

Please note:

- Commercial temperature range is 0°C to 70°C. Extended temperature range is −40°C to +85°C.
- Burn-in is dynamic, for a minimum time of 160 hours at 125°C, V_{CC} = 5.5V ±0.25V, following guidelines in MIL-STD-883 Method 1015 (Test Condition D).
- The following devices are not available in ceramic packages:
  8051AH, 8031AH
  8052AH, 8032AH
- The following devices are not available in extended temperature range:
  8751H, 8751H-8

Examples: P8031AH indicates 8031AH in a plastic package and specified for commercial temperature range, without burn-in. LD8051AH indicates 8051AH in a cerdip package and specified for extended temperature range with burn-in.

8751BH
SINGLE-CHIP 8-BIT MICROCOMPUTER WITH 4K BYTES OF EPROM PROGRAM MEMORY

- Program Memory Lock
- 128 Bytes Data Ram
- Quick Pulse Programming™ Algorithm
- 12.75 Volt Programming Voltage
- Boolean Processor
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- 5 Interrupt Sources
- Programmable Serial Channel
- 64K External Program Memory Space
- 64K External Data Memory Space

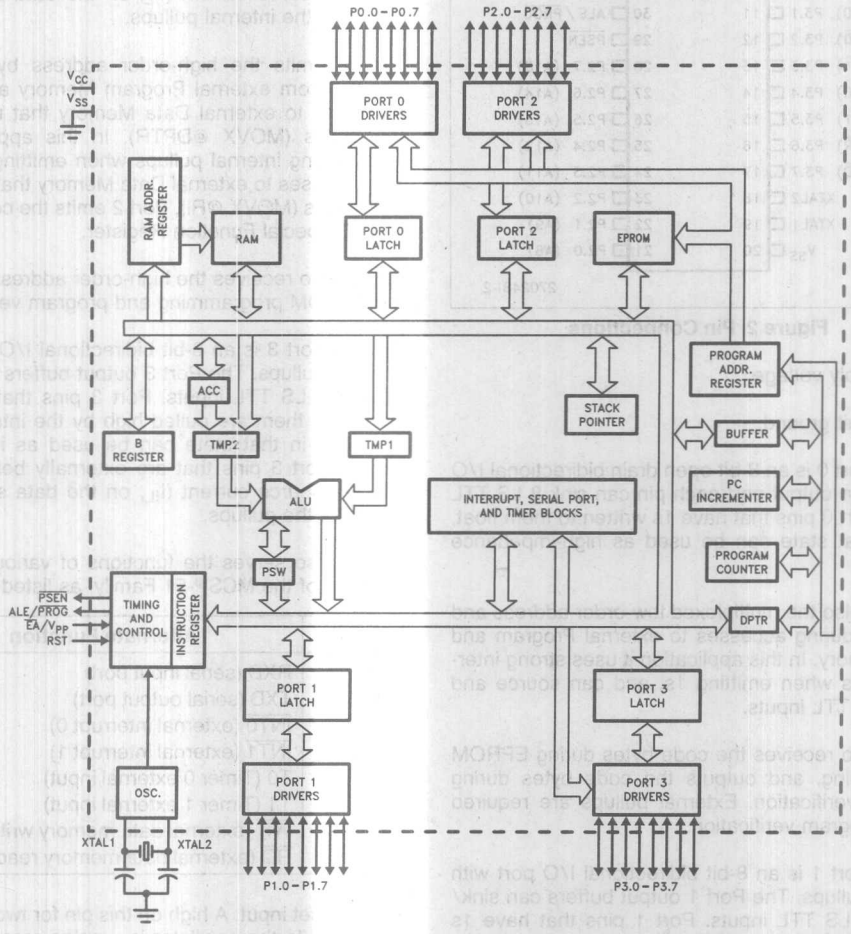


Figure 1. 8751BH Block Diagram

August 1987
Order Number: 270248-002

PIN DESCRIPTIONS

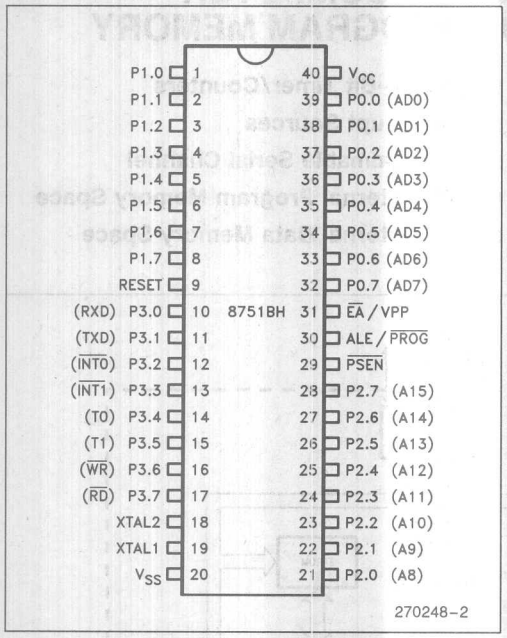


Figure 2. Pin Connections

V_{CC}: Supply voltage.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s, and can source and sink 8 LS TTL inputs.

Port 0 also receives the code bytes during EPROM programming, and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during EPROM programming and program verification.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits during EPROM programming and program verification.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS®-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during EPROM programming.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

$\overline{PSEN}$: Program Store Enable is the Read strobe to External Program Memory.

When the 8751BH is executing code from external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to External Data Memory.

$\overline{EA}$/V_{PP}: External Access enable. $\overline{EA}$ must be strapped to V_{SS} in order to enable the device to fetch code from External Program Memory locations 0000H to 0FFFH. Note, however, that if either of the Lock Bits are programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Applications Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

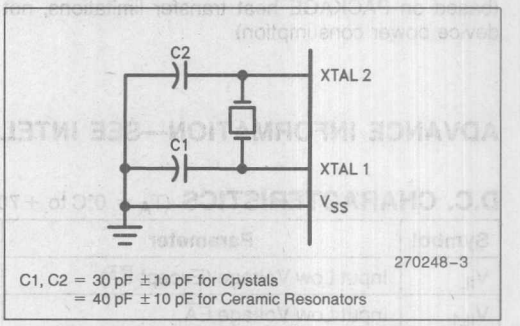


Figure 3. Oscillator Connections

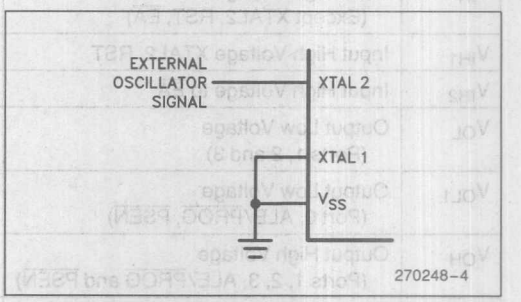


Figure 4. External Clock Drive Configuration

DESIGN CONSIDERATIONS

Exposure to light when the device is in operation may cause logic errors. For this reason, it is suggested that an opaque label be placed over the window when the die is exposed to ambient light.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on $\overline{EA}$/V_{PP} Pin to V_{SS} . . . −0.5V to +13.0V
Voltage on Any Other Pin to V_{SS} . . . . −0.5V to +7V
Power Dissipation 1.5W
(based on PACKAGE heat transfer limitations, not device power consumption)

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

ADVANCE INFORMATION—SEE INTEL FOR DESIGN-IN INFORMATION

D.C. CHARACTERISTICS (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V)

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage (Except $\overline{EA}$)	−0.5	0.8	V	
V_{IL1}	Input Low Voltage $\overline{EA}$	V_{SS}	0.7	V	
V_{IH}	Input High Voltage (Except XTAL2, RST, $\overline{EA}$)	2.0	V_{CC}+0.5	V	
V_{IH1}	Input High Voltage XTAL2, RST	2.5	V_{CC}+0.5	V	XTAL1 = VSS
V_{IH2}	Input High Voltage to $\overline{EA}$	4.5	5.5	V	
V_{OL}	Output Low Voltage (Ports 1, 2 and 3)		0.45	V	I_{OL} = 1.6 mA (Note 1)
V_{OL1}	Output Low Voltage (Port 0, ALE/$\overline{PROG}$, $\overline{PSEN}$)		0.45	V	I_{OL} = 3.2 mA (Notes 1, 2)
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE/$\overline{PROG}$ and $\overline{PSEN}$)	2.4		V	I_{OH} = −80 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4		V	I_{OH} = −400 μA
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3 and RST)		−1	mA	V_{IN} = 0.45V
I_{IL1}	Logical 0 Input Current ($\overline{EA}$)		−10	mA	V_{IN} = VSS
I_{IL2}	Logical 0 Input Current (XTAL2)		−3.2	mA	V_{IN} = 0.45 V XTAL1 = V_{SS}
I_{LI}	Input Leakage Current (Port 0)		±10	μA	0.45 < V_{IN} < V_{CC}
I_{IH}	Logical 1 Input Current ($\overline{EA}$)		1	mA	4.5V < V_{IN} < 5.5V
I_{IH1}	Input Current to RST to Activate Reset		500	μA	V_{IN} < (V_{CC} − 1.5V)
I_{CC}	Power Supply Current		175	mA	All Outputs Disconnected
C_{IO}	Pin Capacitance		10	pF	Test Freq = 1MHz

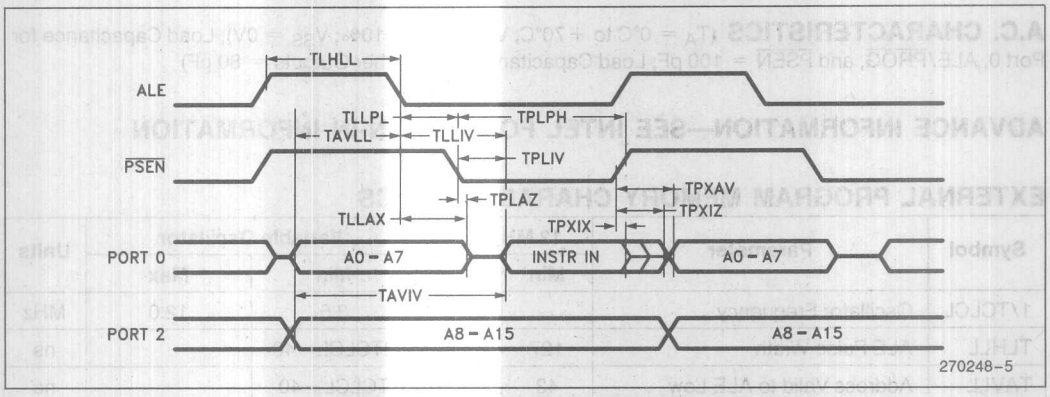
NOTES:
1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE/$\overline{PROG}$ and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE/$\overline{PROG}$ pin may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
2. ALE/$\overline{PROG}$ refers to a pin on the 8751BH. ALE refers to a timing signal that is output on the ALE/$\overline{PROG}$ pin.

A.C. CHARACTERISTICS (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V); Load Capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; Load Capacitance for All Other Outputs = 80 pF)

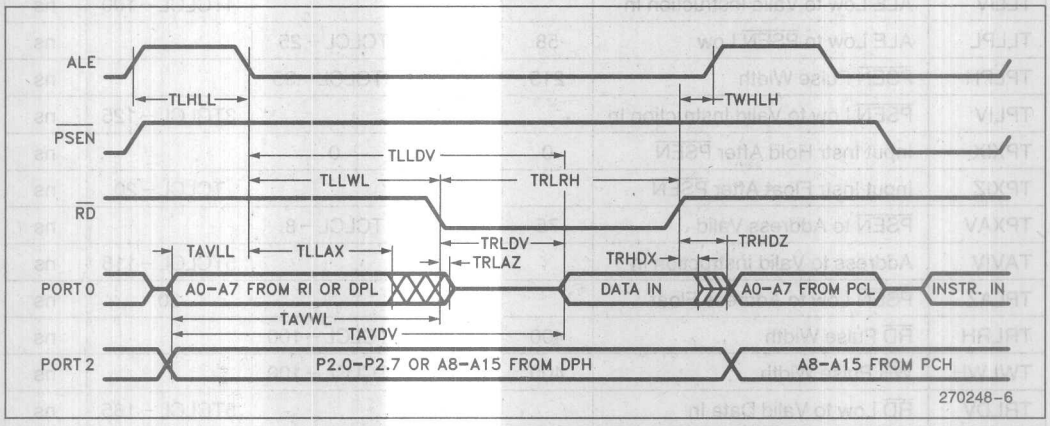
ADVANCE INFORMATION—SEE INTEL FOR DESIGN-IN INFORMATION

EXTERNAL PROGRAM MEMORY CHARACTERISTICS

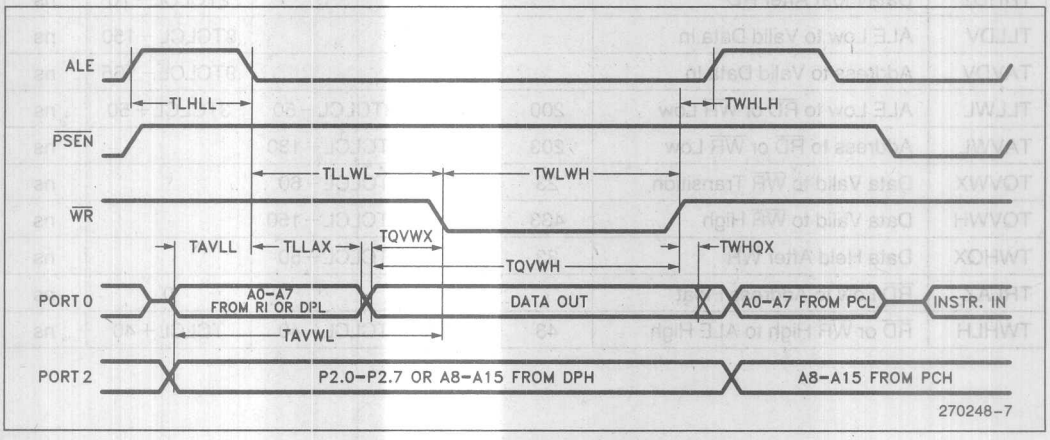
Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL−40		ns
TLLAX	Address Hold After ALE Low	48		TCLCL−35		ns
TLLIV	ALE Low to Valid Instruction In		233		4TCLCL−100	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	58		TCLCL−25		ns
TPLPH	$\overline{PSEN}$ Pulse Width	215		3TCLCL−35		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instruction In		125		3TCLCL−125	ns
TPXIX	Input Instr Hold After $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float After $\overline{PSEN}$		63		TCLCL−20	ns
TPXAV	$\overline{PSEN}$ to Address Valid	75		TCLCL−8		ns
TAVIV	Address to Valid Instruction In		302		5TCLCL−115	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		20		20	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL−100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL−100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		252		5TCLCL−165	ns
TRHDX	Data Hold After $\overline{RD}$	0		0		ns
TRHDZ	Data Float After $\overline{RD}$		97		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL−150	ns
TAVDV	Address to Valid Data In		585		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition	23		TCLCL−60		ns
TQVWH	Data Valid to $\overline{WR}$ High	433		7TCLCL−150		ns
TWHQX	Data Held After $\overline{WR}$	33		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	TCLCL−40	TCLCL+40	ns



External Program Memory Read Cycle



External Data Memory Read Cycle

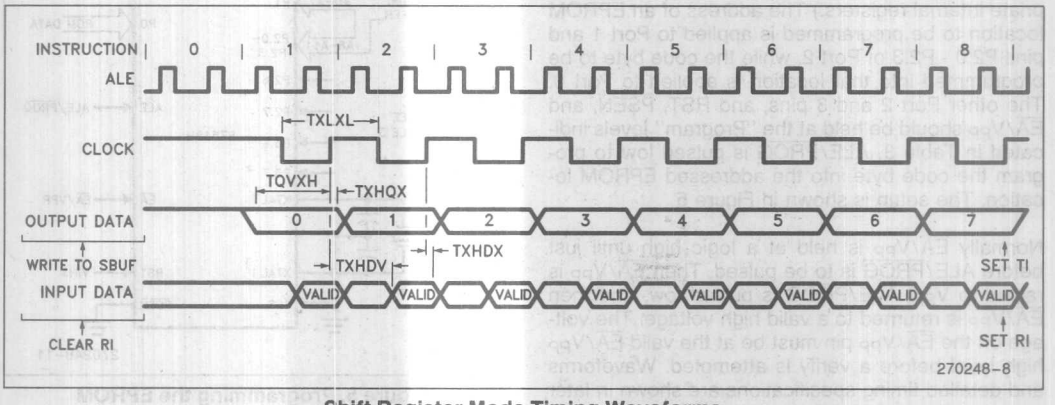


External Data Memory Write Cycle

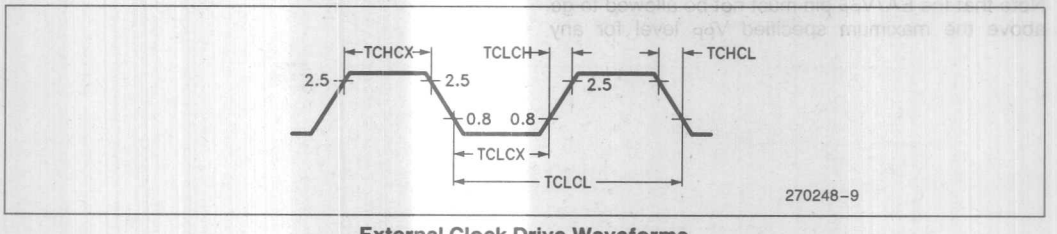
SERIAL PORT TIMING — SHIFT REGISTER MODE

TEST CONDITIONS (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance = 80 pF)

Symbol	Parameter	12MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL−133		ns
TXHQX	Output Data Hold After Clock Rising Edge	50		2TCLCL−117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL−133	ns



Shift Register Mode Timing Waveforms

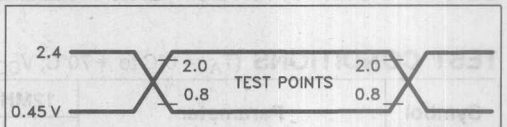


External Clock Drive Waveforms

EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

AC TESTING INPUT/OUTPUT WAVEFORMS



AC inputs during testing are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".

EPROM CHARACTERISTICS

Programming the EPROM

To be programmed, the part must be running with a 4 to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate internal registers.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0 - P2.3 of Port 2, while the code byte to be programmed into that location is applied to Port 0. The other Port 2 and 3 pins, and RST, $\overline{PSEN}$, and $\overline{EA}/V_{PP}$ should be held at the "Program" levels indicated in Table 3. ALE/$\overline{PROG}$ is pulsed low to program the code byte into the addressed EPROM location. The setup is shown in Figure 5.

Normally $\overline{EA}/V_{PP}$ is held at a logic high until just before ALE/$\overline{PROG}$ is to be pulsed. Then $\overline{EA}/V_{PP}$ is raised to V_{PP}, ALE/$\overline{PROG}$ is pulsed low, and then $\overline{EA}/V_{PP}$ is returned to a valid high voltage. The voltage on the $\overline{EA}/V_{PP}$ pin must be at the valid $\overline{EA}/V_{PP}$ high level before a verify is attempted. Waveforms and detailed timing specifications are shown in later sections of this data sheet.

Note that the $\overline{EA}/V_{PP}$ pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches.

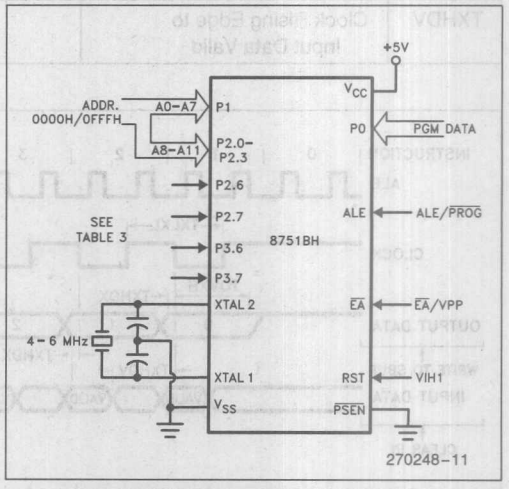


Figure 5. Programming the EPROM

Table 3. EPROM Programming Modes

MODE	RST	PSEN	ALE/PROG	EA/VPP	P2.7	P2.6	P3.6	P3.7
Program Code Data	1	0	0*	VPP	1	0	1	1
Verify Code Data	1	0	1	1	0	0	1	1
Program Encryption Table Use Addresses 0-1FH	1	0	0*	VPP	1	0	0	1
Program Lock x=1	1	0	0*	VPP	1	1	1	1
Bits (LBx) x=2	1	0	0*	VPP	1	1	0	0

NOTES:
"1" = Valid high for that pin
"0" = Valid low for that pin
"VPP" = +12.75V ±0.25V
* ALE/PROG is pulsed low for 100 uS for programming. (Quick-Pulse Programming™)

QUICK-PULSE PROGRAMMING™ ALGORITHM

The 8751BH can be programmed using the Quick-Pulse Programming Algorithm for microcontrollers. The features of the new programming method are a lower VPP (12.75 volts as compared to 21 volts) and a shorter programming pulse. It is possible to program the entire 4K Bytes of EPROM memory in less than 13 seconds with this algorithm

To program the part using the new algorithm, VPP must be 12.75 ±0.25 Volts. ALE/PROG is pulsed low for 100 μseconds, 25 times. Then, the byte just programmed may be verified. After programming, the entire array should be verified. The Program Lock features are programmed using the same method, but with the setup as shown in Table 3. The only difference in programming Lock features is that the Lock features cannot be directly verified. Instead, verification of programming is by observing that their features are enabled.

PROGRAM VERIFICATION

If the Lock Bits have not been programmed, the on-chip Program Memory can be read out for verification purposes, if desired, either during or after the programming operation. The address of the Program Memory location to be read is applied to Port 1 and pins P2.0 - P2.3. The other pins should be held at the "Verify" levels indicated in Table 3. The contents of the addressed location will come out on Port 0. External pullups are required on Port 0 for this operation. (If the Encryption Array in the EPROM has been programmed, the data present at Port 0 will be Code Data XNOR Encryption Data. The user must know the Encryption Array contents to manually "unencrypt" the data during verify.)

The setup, which is shown in Figure 6, is the same as for programming the EPROM except that pin P2.7 is held at a logic low, or may be used as an active low read strobe.

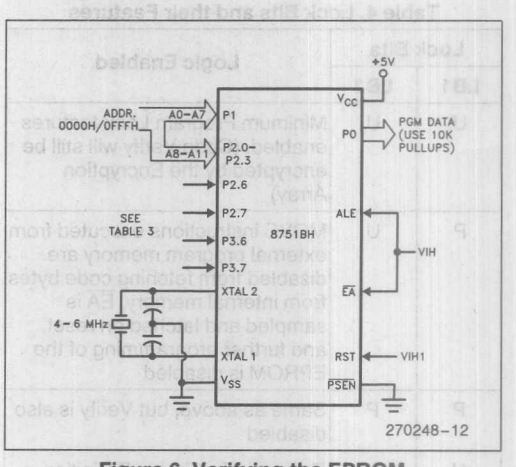


Figure 6. Verifying the EPROM

PROGRAM MEMORY LOCK

The two-level Program Lock system consists of 2 Lock bits and a 32-byte Encryption Array which are used to protect the program memory against software piracy.

ENCRYPTION ARRAY

Within the EPROM array are 32 bytes of Encryption Array that are initially unprogrammed (all 1s). Every time that a byte is addressed during a verify, 5 address lines are used to select a byte of the Encryption Array. This byte is then exclusive-NORed (XNOR) with the code byte, creating an Encrypted Verify byte. The algorithm, with the array in the unprogrammed state (all 1s), will return the code in its original, unmodified form.

It is recommended that whenever the Encryption Array is used, at least one of the Lock Bits be programmed as well.

LOCK BITS

Also included in the EPROM Program Lock scheme are two Lock Bits which function as shown in Table 4.

Table 4. Lock Bits and their Features

Lock Bits		Logic Enabled
LB1	LB2	
U	U	Minimum Program Lock features enabled. (Code Verify will still be encrypted by the Encryption Array)
P	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the EPROM is disabled
P	P	Same as above, but Verify is also disabled
U	P	Reserved for Future Definition

P = Programmed
U = Unprogrammed

Erasing the EPROM also erases the Encryption Array and the Lock Bits, returning the part to full unlocked functionality.

To ensure proper functionality of the chip, the internally latched value of the $\overline{EA}$ pin must agree with its external state.

ERASURE CHARACTERISTICS

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at lease 15 W-sec/cm. Exposing the EPROM to an ultraviolet lamp of 12,000 μW/cm rating for 20 to 30 minutes, at a distance of about 1 inch, should be sufficient.

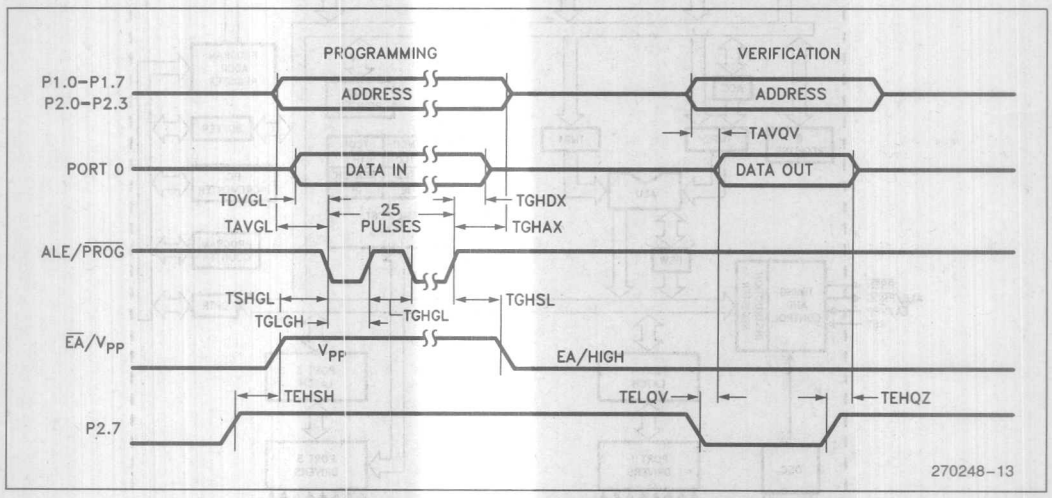
Erasure leaves the array in an all 1s state.

ADVANCE INFORMATION—SEE INTEL FOR DESIGN-IN INFORMATION

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

(T_A = 21°C to 27°C, V_{CC} = 5.0V ±10%, V_{SS} = 0V)

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	12.5	13.0	V
IPP	Programming Supply Current		50	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{PROG}$ Low	48TCLCL		
TGHAX	Address Hold After $\overline{PROG}$	48TCLCL		
TDVGL	Data Setup to $\overline{PROG}$ Low	48TCLCL		
TGHDX	Data Hold After $\overline{PROG}$	48TCLCL		
TEHSH	P2.7 ($\overline{ENABLE}$) High to V_{PP}	48TCLCL		
TSHGL	V_{PP} Setup to $\overline{PROG}$ Low	10		µsec
TGHSL	V_{PP} Hold After $\overline{PROG}$	10		µsec
TGLGH	$\overline{PROG}$ Width	90	110	µsec
TAVQV	Address to Data Valid		48TCLCL	
TELQV	$\overline{ENABLE}$ Low to Data Valid		48TCLCL	
TEHQZ	Data Float After $\overline{ENABLE}$	0	48TCLCL	
TGHGL	$\overline{PROG}$ High to $\overline{PROG}$ Low	10		µsec



EPROM Programming and Verification Waveforms

8052BH
SINGLE-CHIP 8-BIT MICROCOMPUTER WITH FACTORY MASK-PROGRAMMABLE ROM
8032BH
SINGLE-CHIP 8-BIT CONTROL-ORIENTED CPU WITH RAM AND I/O

8032BH—ROMless

8052BH—8K Bytes of Factory Mask-Programmed ROM

- 256 Bytes Data Ram
- Boolean Processor
- 32 Programmable I/O Lines
- Three 16-Bit Timer/Counters
- 6 Interrupt Sources
- Programmable Serial Channel
- Separate Transmit/Receive Baud Rate Capability
- 64K External Program Memory Space
- 64K External Data Memory Space

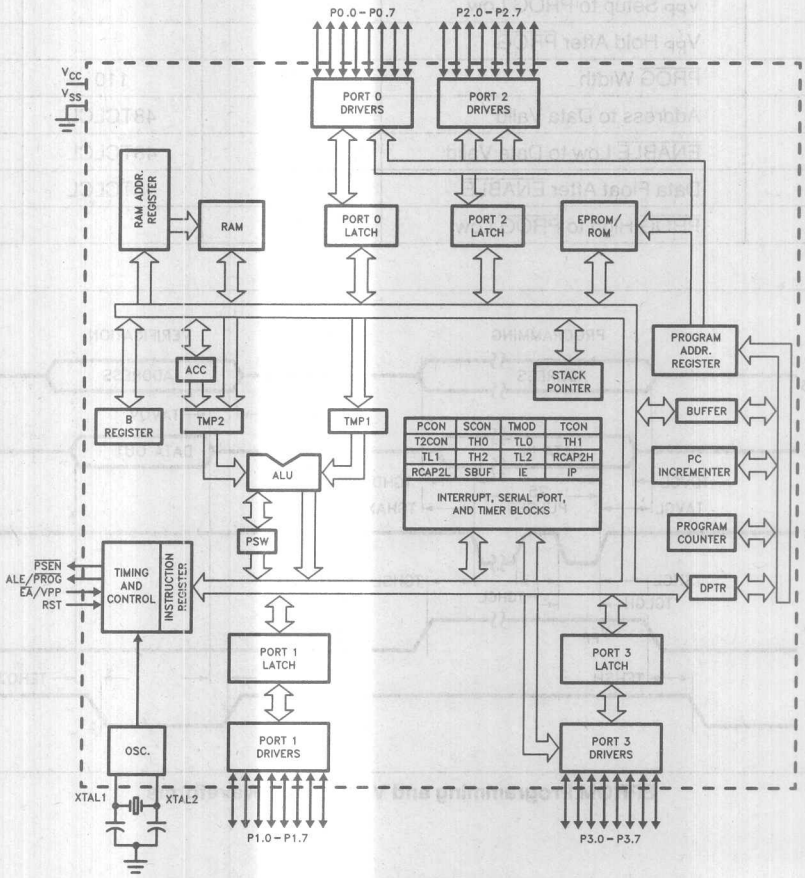


Figure 1. Block Diagram

270192–1

October 1987
Order Number: 270192-003

PIN DESCRIPTIONS

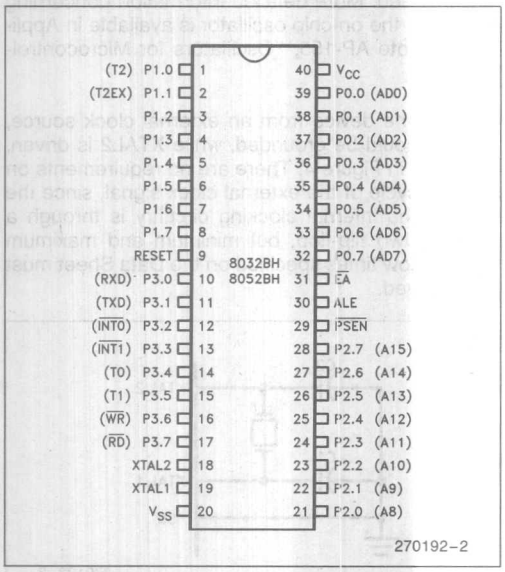


Figure 2. Pin Connections

V_{CC}: Supply voltage.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s, and can source and sink 8 LS TTL inputs.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

In addition, P1.0 and P1.1 serve the functions of the following special features of the MCS®-51 Family:

Port Pin	Alternate Function
P1.0	T2 (Timer/Counter 2 External Input)
P1.1	T2EX (Timer/Counter 2 Capture/Reload Trigger)

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS®-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

$\overline{PSEN}$: Program Store Enable is the Read strobe to External Program Memory.

When the device is executing code from external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to External Data Memory.

$\overline{EA}$: External Access enable. $\overline{EA}$ must be strapped to V_{SS} in order to enable the device to fetch code from External Program Memory locations 0000H to 1FFFH. Note, however, that if either of the Lock Bits are programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program executions.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Applications Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

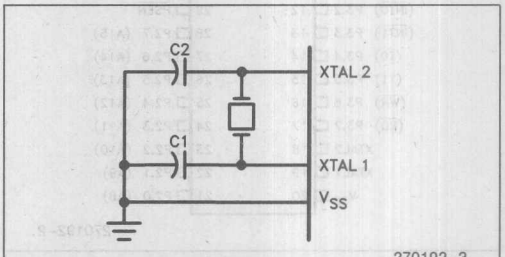


Figure 3. Oscillator Connections

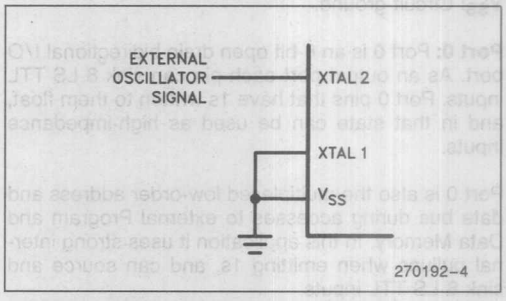


Figure 4. External Clock Drive Configuration

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on $\overline{EA}$ Pin to V_{SS} . . . . . −0.5V to +13.0V
Voltage on Any Other Pin to V_{SS} . . . . −0.5V to +7V
Power Dissipation 1.5W
(based on PACKAGE heat transfer limitations, not device power consumption)

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

ADVANCE INFORMATION. Contact Intel for Design-In Information.

D.C. CHARACTERISTICS (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V)

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage (Except $\overline{EA}$)	−0.5	0.8	V	
V_{IL1}	Input Low Voltage $\overline{EA}$	V_{SS}	0.7	V	
V_{IH}	Input High Voltage (Except XTAL2, RST, $\overline{EA}$)	2.0	V_{CC}+0.5	V	
V_{IH1}	Input High Voltage XTAL2, RST	2.5	V_{CC}+0.5	V	XTAL1 = V_{SS}
V_{IH2}	Input High Voltage to $\overline{EA}$	4.5	5.5	V	
V_{OL}	Output Low Voltage (Ports 1, 2 and 3)		0.45	V	I_{OL} = 1.6 mA (Note 1)
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$)		0.45	V	I_{OL} = 3.2 mA (Note 1)
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE and $\overline{PSEN}$)	2.4		V	I_{OH} = −80 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4		V	I_{OH} = −400 μA
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3 and RST)		−500	μA	V_{IN} = 0.45 V
I_{IL1}	Logical 0 Input Current ($\overline{EA}$)	−10	500	mA μA	V_{IN} = V_{SS}
I_{IL2}	Logical 0 Input Current (XTAL2)		−3.2	mA	V_{IN} = 0.45V XTAL1 = V_{SS}
I_{LI}	Input Leakage Current (Port 0)		±10	μA	0.45 < V_{IN} < V_{CC}
I_{IH}	Logical 1 Input Current ($\overline{EA}$)		1	mA	4.5V < V_{IN} < 5.5V
I_{IH1}	Input Current to RST to activate Reset		500	μA	V_{IN} < (V_{CC} − 1.5V)
I_{CC}	Power Supply Current		175	mA	All Outputs Disconnected
C_{IO}	Pin Capacitance		10	pF	Test freq = 1MHz

NOTES:
1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

Each timing symbol has 5 characters. The first character is always a "T" (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

A:Address.
C:Clock.
D:Input data.
H:Logic level HIGH.
I:Instruction (program memory contents).
P:$\overline{PSEN}$.
Q:Output data.
R:$\overline{RD}$ signal.
T:Time.
V:Valid.
W:$\overline{WR}$ signal.
X:No longer a valid logic level.
Z:Float.

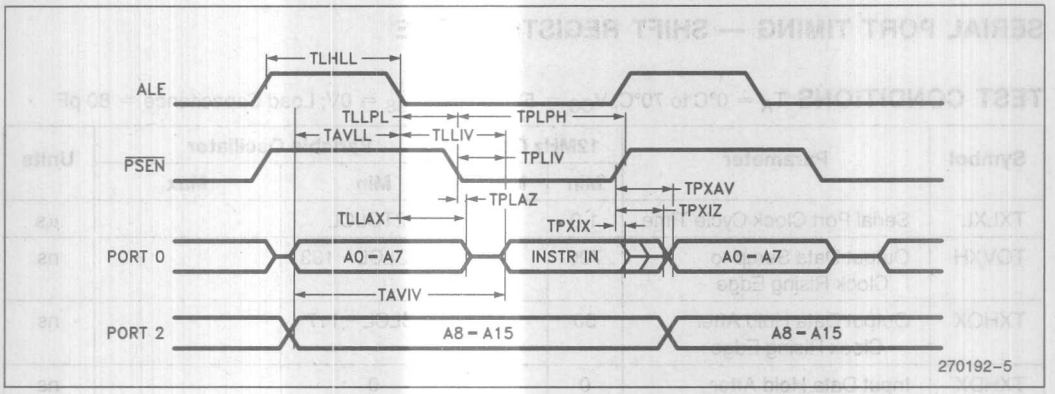
For example,
TAVLL = Time from Address Valid to ALE Low.
TLLPL = Time from ALE Low to PSEN Low.

A.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ±10%; V_{SS} = 0V); Load Capacitance for Port 0, ALE and $\overline{PSEN}$ = 100 pF; Load Capacitance for All Other Outputs = 80 pF)

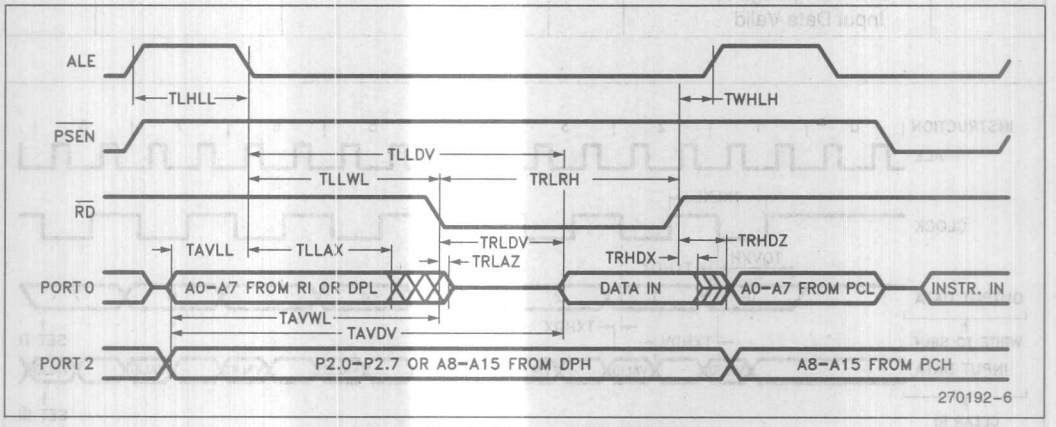
ADVANCE INFORMATION. Contact Intel for Design-In Information.

EXTERNAL PROGRAM MEMORY CHARACTERISTICS

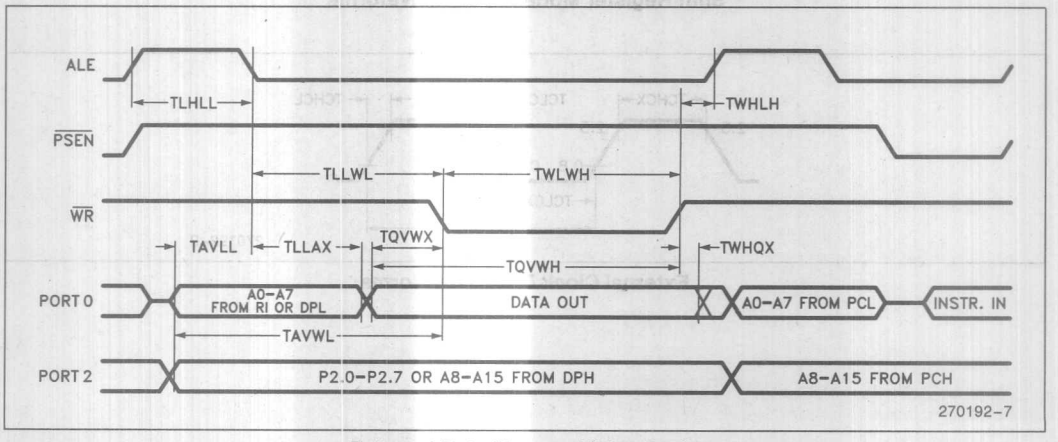
Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL−40		ns
TLLAX	Address Hold After ALE Low	48		TCLCL−35		ns
TLLIV	ALE Low to Valid Instruction In		233		4TCLCL−100	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	58		TCLCL−25		ns
TPLPH	$\overline{PSEN}$ Pulse Width	215		3TCLCL−35		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instruction In		125		3TCLCL−125	ns
TPXIX	Input Instr Hold After $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float After $\overline{PSEN}$		63		TCLCL−20	ns
TPXAV	$\overline{PSEN}$ to Address Valid	75		TCLCL−8		ns
TAVIV	Address to Valid Instruction In		302		5TCLCL−115	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		20		20	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL−100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL−100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		252		5TCLCL−165	ns
TRHDX	Data Hold After $\overline{RD}$	0		0		ns
TRHDZ	Data Float After $\overline{RD}$		97		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL−150	ns
TAVDV	Address to Valid Data In		585		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition	23		TCLCL−60		ns
TQVWH	Data Valid to $\overline{WR}$ High	433		7TCLCL−150		ns
TWHQX	Data Held After $\overline{WR}$	33		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	TCLCL−40	TCLCL+40	ns



External Program Memory Read Cycle



External Data Memory Read Cycle

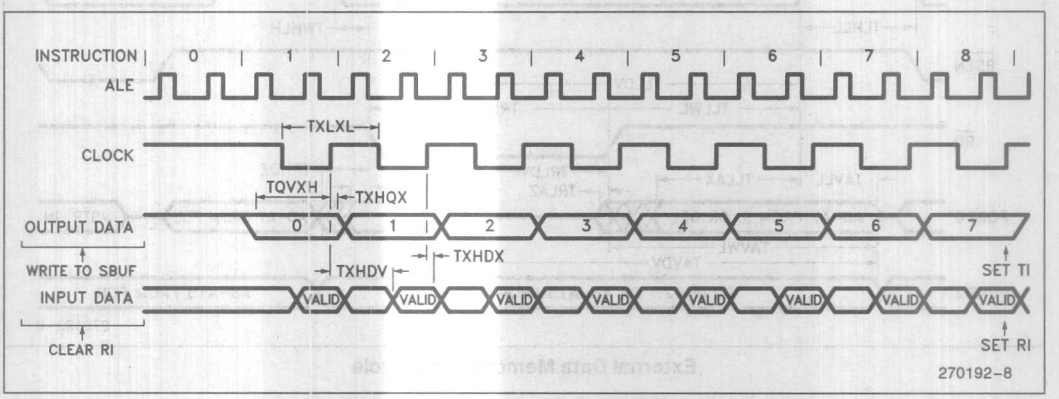


External Data Memory Write Cycle

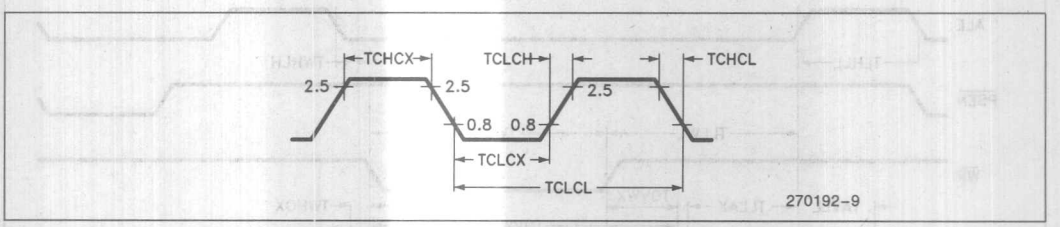
SERIAL PORT TIMING — SHIFT REGISTER MODE

TEST CONDITIONS T_A = 0°C to 70°C; V_{CC} = 5V ± 10%; V_{SS} = 0V; Load Capacitance = 80 pF

Symbol	Parameter	12MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL − 133		ns
TXHQX	Output Data Hold After Clock Rising Edge	50		2TCLCL − 117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL − 133	ns



Shift Register Mode Timing Waveforms



External Clock Drive Waveforms

EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

PROGRAM MEMORY LOCK

The two-level Program Lock system consists of 2 Lock bits and a 32-byte Encryption Array which are used to protect the program memory against software piracy. The following description applies to the 8752BH. The same options are also available on the 8052BH, mask-programmed at the factory.

ENCRYPTION ARRAY

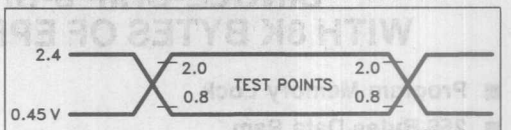
Within the EPROM array are 32 bytes of Encryption Array that are initially unprogrammed (all 1s). Every time that a byte is addressed during a verify, 5 address lines are used to select a byte of the Encryption Array. This byte is then exclusive-NORed (XNOR) with the code byte, creating an Encrypted Verify byte. The algorithm, with the array in the unprogrammed state (all 1s), will return the code in its original, unmodified form.

It is recommended that whenever the Encryption Array is used, at least one of the Lock Bits be programmed as well.

LOCK BITS

Also included in the Program Lock scheme are two Lock Bits which function as shown in Table 1.

AC TESTING INPUT/OUTPUT WAVEFORMS



270192-10

AC inputs during testing are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".

Table 1. Lock Bits and their Features

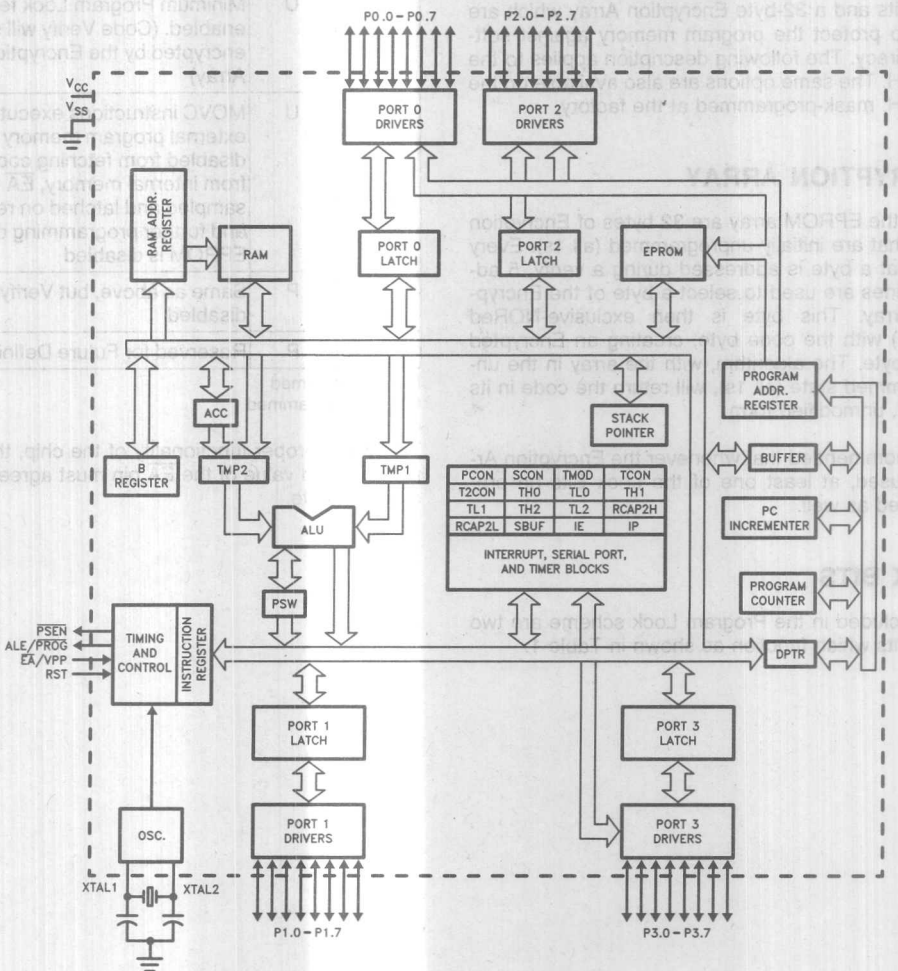
Lock Bits		Logic Enabled
LB1	LB2	
U	U	Minimum Program Lock features enabled. (Code Verify will still be encrypted by the Encryption Array)
P	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the EPROM is disabled
P	P	Same as above, but Verify is also disabled
U	P	Reserved for Future Definition

P = Programmed
U = Unprogrammed

To ensure proper functionality of the chip, the internally latched value of the $\overline{EA}$ pin must agree with its external state.

8752BH
SINGLE-CHIP 8-BIT MICROCOMPUTER WITH 8K BYTES OF EPROM PROGRAM MEMORY

- Program Memory Lock
- 256 Bytes Data Ram
- Quick Pulse Programming™ Algorithm
- 12.75 Volt Programming Voltage
- Boolean Processor
- 32 Programmable I/O Lines
- Three 16-Bit Timer/Counters
- 6 Interrupt Sources
- Programmable Serial Channel
- Separate Transmit/Receive Baud Rate Capability
- 64K External Program Memory Space
- 64K External Data Memory Space



270429-1

Figure 1. Block Diagram

October 1987
Order Number: 270429-001

PIN DESCRIPTIONS

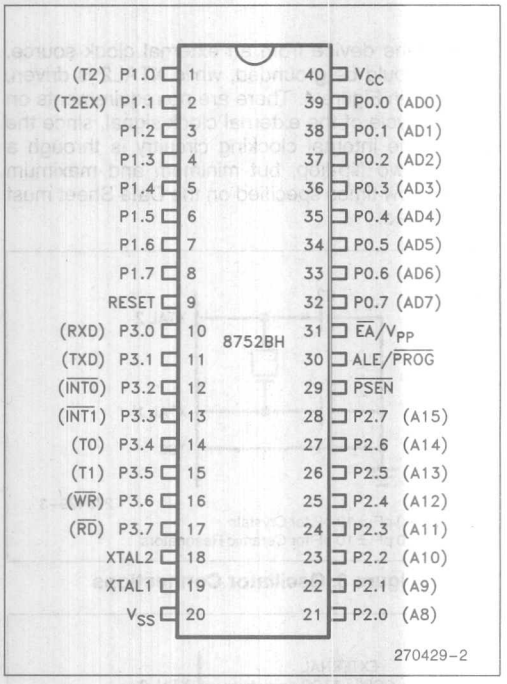


Figure 2. Pin Connections

V_{CC}: Supply voltage.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s, and can source and sink 8 LS TTL inputs.

Port 0 also receives the code bytes during EPROM programming, and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during EPROM programming and program verification.

In addition, P1.0 and P1.1 serve the functions of the following special features of the MCS®-51 Family:

Port Pin	Alternate Function
P1.0	T2 (Timer/Counter 2 External Input)
P1.1	T2EX (Timer/Counter 2 Capture/Reload Trigger)

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits during EPROM programming and program verification.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS®-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during EPROM programming on the 8752BH.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

$\overline{PSEN}$: Program Store Enable is the Read strobe to External Program Memory.

When the device is executing code from external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to External Data Memory.

$\overline{EA}$/V_{PP}: External Access enable. $\overline{EA}$ must be strapped to V_{SS} in order to enable the device to fetch code from External Program Memory locations 0000H to 1FFFH. Note, however, that if either of the Lock Bits are programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

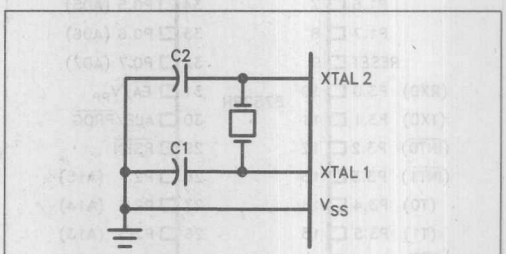


Figure 3. Oscillator Connections

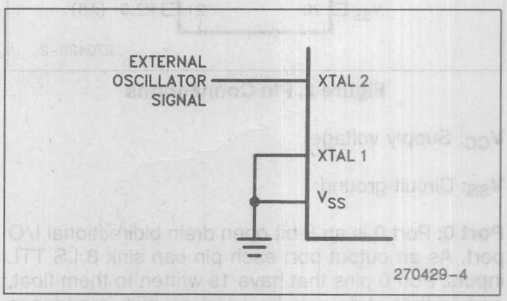


Figure 4. External Clock Drive Configuration

DESIGN CONSIDERATIONS

Exposure to light when the 8752BH is in operation may cause logic errors. For this reason, it is suggested that an opaque label be placed over the window of the 8752BH when the die is exposed to ambient light.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on $\overline{EA}/V_{PP}$ Pin to V_{SS} ...−0.5V to +13.0V
Voltage on Any Other Pin to V_{SS} ....−0.5V to +7V
Power Dissipation.....1.5W
(based on PACKAGE heat transfer limitations, not device power consumption)

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (Except $\overline{EA}$)	−0.5	0.8	V	
V_{IL1}	Input Low Voltage $\overline{EA}$	V_{SS}	0.7	V	
V_{IH}	Input High Voltage (Except XTAL2, RST, $\overline{EA}$)	2.0	V_{CC}+0.5	V	
V_{IH1}	Input High Voltage XTAL2, RST	2.5	V_{CC}+0.5	V	XTAL1 = V_{SS}
V_{IH2}	Input High Voltage to $\overline{EA}$	4.5	5.5	V	
V_{OL}	Output Low Voltage (Ports 1, 2 and 3)		0.45	V	I_{OL} = 1.6 mA (Note 1)
V_{OL1}	Output Low Voltage (Port 0, ALE/$\overline{PROG}$, $\overline{PSEN}$)		0.45	V	I_{OL} = 3.2 mA (Note 1, 2)
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE/$\overline{PROG}$ and $\overline{PSEN}$)	2.4		V	I_{OH} = −80 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4		V	I_{OH} = −400 μA
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3 and RST)		−500	μA	V_{IN} = 0.45V
I_{IL1}	Logical 0 Input Current ($\overline{EA}$)	−10	500	mA μA	V_{IN} = V_{SS}
I_{IL2}	Logical 0 Input Current (XTAL2)		−3.2	mA	V_{IN} = 0.45V XTAL1 = V_{SS}
I_{LI}	Input Leakage Current (Port 0)		±10	μA	0.45 < V_{IN} < V_{CC}
I_{IH}	Logical 1 Input Current ($\overline{EA}$)		1	mA	4.5V < V_{IN} < 5.5V
I_{IH1}	Input Current to RST to activate Reset		500	μA	V_{IN} < (V_{CC} − 1.5V)
I_{CC}	Power Supply Current		175	mA	All Outputs Disconnected
C_{IO}	Pin Capacitance		10	pF	Test freq = 1 MHz

NOTES:
1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE/$\overline{PROG}$ and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE/$\overline{PROG}$ pin may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
2. ALE/$\overline{PROG}$ refers to a pin on the device. ALE refers to a timing signal that is output on the ALE/$\overline{PROG}$ pin.

Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

A:Address
C:Clock
D:Input Data
H:Logic level HIGH
I:Instruction (program memory contents)
P:PSEN
Q:Output data
R:$\overline{RD}$ signal
T:Time
V:Valid
W:$\overline{WR}$ signal
X:No longer a valid logic level
Z:Float

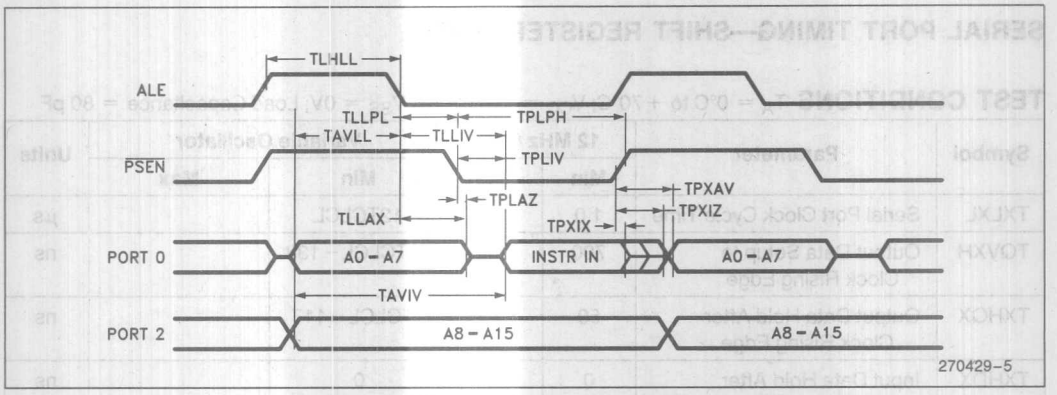
For example,

TAVLL = Time from Address Valid to ALE Low.
TLLPL = Time from ALE Low to PSEN Low.

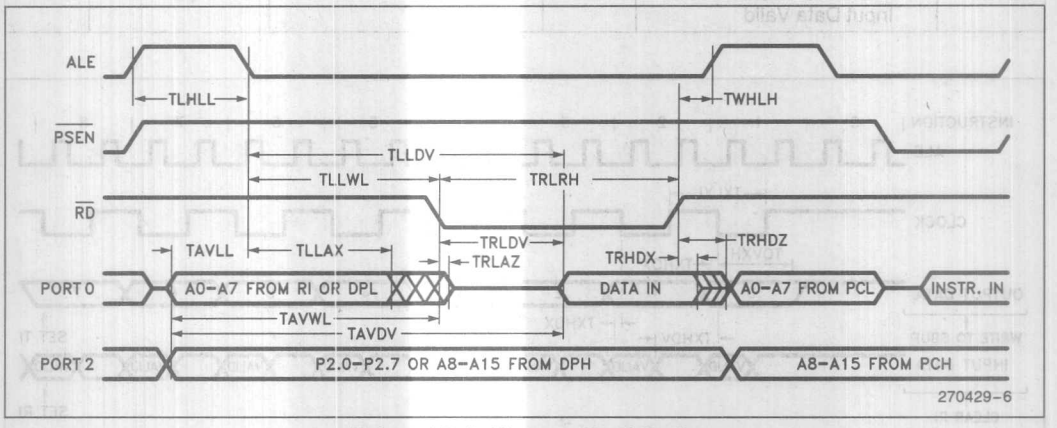
A.C. CHARACTERISTICS (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V); Load Capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; Load Capacitance for All Other Outputs = 80 pF)

EXTERNAL PROGRAM MEMORY CHARACTERISTICS

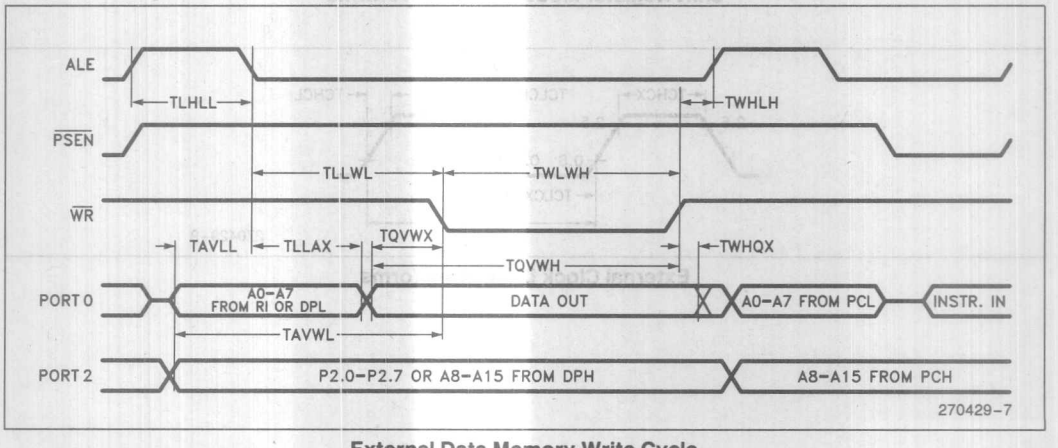
Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL−40		ns
TLLAX	Address Hold After ALE Low	48		TCLCL−35		ns
TLLIV	ALE Low to Valid Instruction In		233		4TCLCL−100	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	58		TCLCL−25		ns
TPLPH	$\overline{PSEN}$ Pulse Width	215		3TCLCL−35		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instruction In		125		3TCLCL−125	ns
TPXIX	Input Instr Hold After $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float After $\overline{PSEN}$		63		TCLCL−20	ns
TPXAV	$\overline{PSEN}$ to Address Valid	75		TCLCL−8		ns
TAVIV	Address to Valid Instruction In		302		5TCLCL−115	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		20		20	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL−100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL−100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		252		5TCLCL−165	ns
TRHDX	Data Hold After $\overline{RD}$	0		0		ns
TRHDZ	Data Float After $\overline{RD}$		97		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL−150	ns
TAVDV	Address to Valid Data In		585		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition	23		TCLCL−60		ns
TQVWH	Data Valid to $\overline{WR}$ High	433		7TCLCL−150		ns
TWHQX	Data Held After $\overline{WR}$	33		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	TCLCL−40	TCLCL+40	ns



External Program Memory Read Cycle



External Data Memory Read Cycle

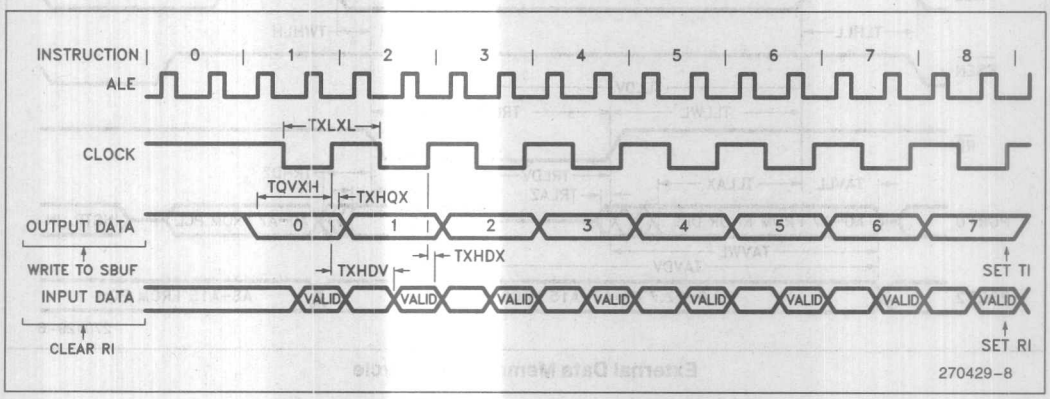


External Data Memory Write Cycle

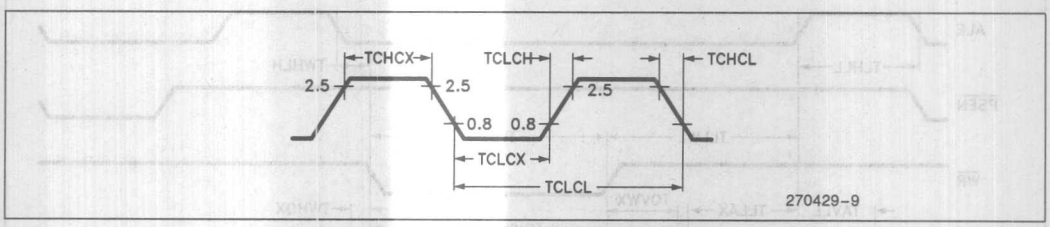
SERIAL PORT TIMING--SHIFT REGISTER MODE

TEST CONDITIONS T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance = 80 pF

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL−133		ns
TXHQX	Output Data Hold After Clock Rising Edge	50		2TCLCL−117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL−133	ns



Shift Register Mode Timing Waveforms

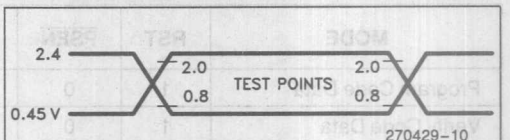


External Clock Drive Waveforms

EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

A.C. TESTING INPUT/OUTPUT WAVEFORMS



AC inputs during testing are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".

EPROM CHARACTERISTICS

Table 1 shows the logic levels for programming the Program Memory, the Encryption Table, and the Lock Bits and for reading the signature bytes.

Programming the EPROM

To be programmed, the 8752BH must be running with a 4 to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate internal registers.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0 - P2.4 of Port 2, while the code byte to be programmed into that location is applied to Port 0. The other Port 2 and 3 pins, and RST, $\overline{PSEN}$, and $\overline{EA}/V_{PP}$ should be held at the "Program" levels indicated in Table 1. ALE/$\overline{PROG}$ is pulsed low to program the code byte into the addressed EPROM location. The setup is shown in Figure 5.

Normally $\overline{EA}/V_{PP}$ is held at a logic high until just before ALE/$\overline{PROG}$ is to be pulsed. Then $\overline{EA}/V_{PP}$ is raised to V_{PP}, ALE/$\overline{PROG}$ is pulsed low, and then $\overline{EA}/V_{PP}$ is returned to a valid high voltage. The voltage on the $\overline{EA}/V_{PP}$ pin must be at the valid $\overline{EA}/V_{PP}$ high level before a verify is attempted. Waveforms and detailed timing specifications are shown in later sections of this data sheet.

Note that the $\overline{EA}/V_{PP}$ pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches.

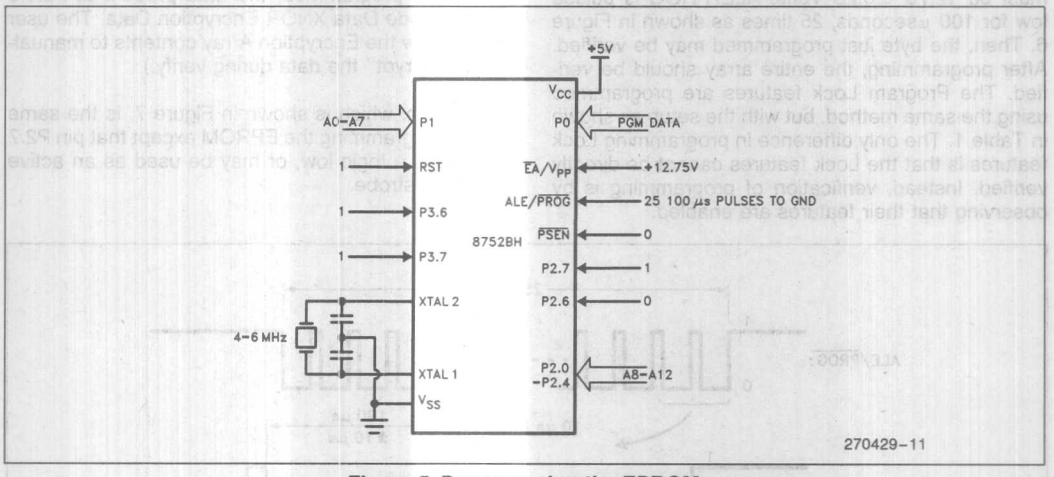


Figure 5. Programming the EPROM

Table 1. EPROM Programming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.6	P3.7
Program Code Data	1	0	0*	V_{PP}	1	0	1	1
Verify Code Data	1	0	1	1	0	0	1	1
Program Encryption Table Use Addresses 0-1FH	1	0	0*	V_{PP}	1	0	0	1
Program Lock Bits (LBx) x=1	1	0	0*	V_{PP}	1	1	1	1
Program Lock Bits (LBx) x=2	1	0	0*	V_{PP}	1	1	0	0
Read Signature	1	0	1	1	0	0	0	0

NOTES:
"1" = Valid high for that pin
"0" = Valid low for that pin
"V_{PP}" = +12.75V ±0.25V
*ALE/$\overline{PROG}$ is pulsed low for 100 uS for programming. (Quick-Pulse Programming™)

QUICK-PULSE PROGRAMMING™ ALGORITHM

The 8752BH can be programmed using the Quick-Pulse Programming™ Algorithm for microcontrollers. The features of the new programming method are a lower V_{PP} (12.75 volts as compared to 21 volts) and a shorter programming pulse. It is possible to program the entire 8K Bytes of EPROM memory in less than 25 seconds with this algorithm!

To program the part using the new algorithm, V_{PP} must be 12.75 ±0.25 Volts. ALE/$\overline{PROG}$ is pulsed low for 100 μseconds, 25 times as shown in Figure 6. Then, the byte just programmed may be verified. After programming, the entire array should be verified. The Program Lock features are programmed using the same method, but with the setup as shown in Table 1. The only difference in programming Lock features is that the Lock features cannot be directly verified. Instead, verification of programming is by observing that their features are enabled.

PROGRAM VERIFICATION

If the Lock Bits have not been programmed, the on-chip Program Memory can be read out for verification purposes, if desired, either during or after the programming operation. The address of the Program Memory location to be read is applied to Port 1 and pins P2.0 - P2.4. The other pins should be held at the "Verify" levels indicated in Table 1. The contents of the addressed location will come out on Port 0. External pullups are required on Port 0 for this operation. (If the Encryption Array in the EPROM has been programmed, the data present at Port 0 will be Code Data XNOR Encryption Data. The user must know the Encryption Array contents to manually "unencrypt" the data during verify.)

The setup, which is shown in Figure 7, is the same as for programming the EPROM except that pin P2.7 is held at a logic low, or may be used as an active low read strobe.

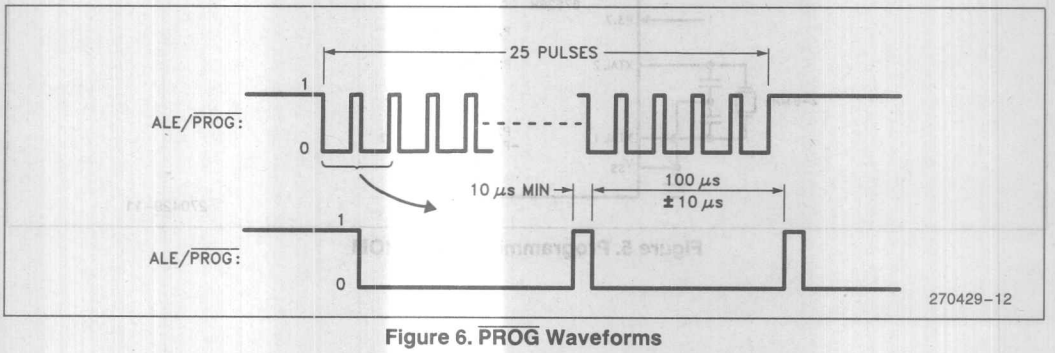


Figure 6. $\overline{PROG}$ Waveforms

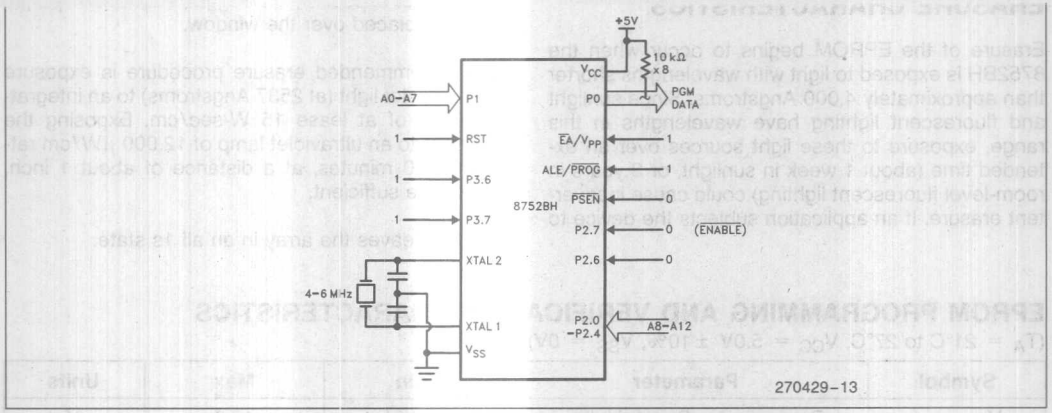


Figure 7. Verifying the EPROM

PROGRAM MEMORY LOCK

The two-level Program Lock system consists of 2 Lock bits and a 32-byte Encryption Array which are used to protect the program memory against software piracy.

ENCRYPTION ARRAY

Within the EPROM array are 32 bytes of Encryption Array that are initially unprogrammed (all 1s). Every time that a byte is addressed during a verify, 5 address lines are used to select a byte of the Encryption Array. This byte is then exclusive-NORed (XNOR) with the code byte, creating an Encrypted Verify byte. The algorithm, with the array in the unprogrammed state (all 1s), will return the code in its original, unmodified form.

It is recommended that whenever the Encryption Array is used, at least one of the Lock Bits be programmed as well.

LOCK BITS

Also included in the EPROM Program Lock scheme are two Lock Bits which function as shown in Table 2.

Erasing the EPROM also erases the Encryption Array and the Lock Bits, returning the part to full unlocked functionality.

To ensure proper functionality of the chip, the internally latched value of the $\overline{EA}$ pin must agree with its external state.

Table 2. Lock Bits and their Features

Lock Bits		Logic Enabled
LB1	LB2	
U	U	Minimum Program Lock features enabled. (Code Verify will still be encrypted by the Encryption Array)
P	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the EPROM is disabled
P	P	Same as above, but Verify is also disabled
U	P	Reserved for Future Definition

P = Programmed
U = Unprogrammed

READING THE SIGNATURE BYTES

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values returned are:

(030H) = 86H indicates manufactured by Intel
(031H) = 52H indicates 8752BH

ERASURE CHARACTERISTICS

Erasure of the EPROM begins to occur when the 8752BH is exposed to light with wavelengths shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

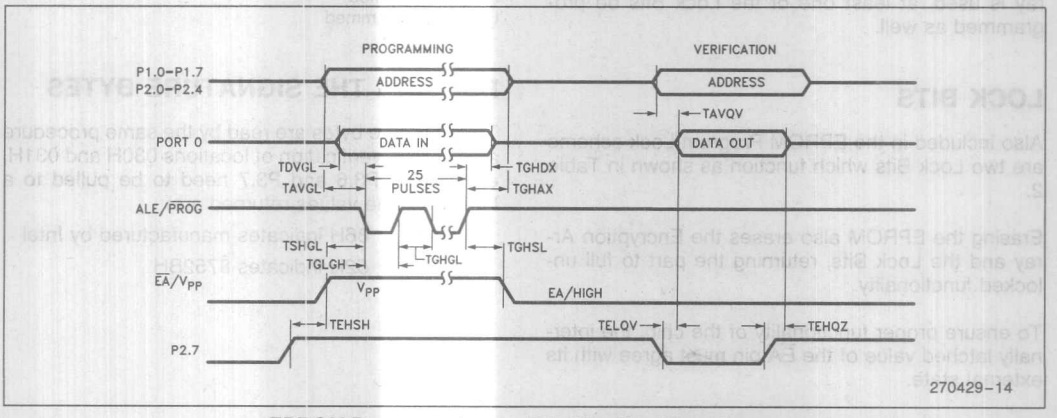
The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at lease 15 W-sec/cm. Exposing the EPROM to an ultraviolet lamp of 12,000 μW/cm rating for 30 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

(T_A = 21°C to 27°C, V_{CC} = 5.0V ±10%, V_{SS} = 0V)

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	12.5	13.0	V
I_{PP}	Programming Supply Current		50	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{PROG}$ Low	48TCLCL		
TGHAX	Address Hold After $\overline{PROG}$	48TCLCL		
TDVGL	Data Setup to $\overline{PROG}$ Low	48TCLCL		
TGHDX	Data Hold After $\overline{PROG}$	48TCLCL		
TEHSH	P2.7 ($\overline{ENABLE}$) High to V_{PP}	48TCLCL		
TSHGL	V_{PP} Setup to $\overline{PROG}$ Low	10		μs
TGHSL	V_{PP} Hold After $\overline{PROG}$	10		μs
TGLGH	$\overline{PROG}$ Width	90	110	μs
TAVQV	Address to Data Valid		48TCLCL	
TELQV	$\overline{ENABLE}$ Low to Data Valid		48TCLCL	
TEHQZ	Data Float After $\overline{ENABLE}$	0	48TCLCL	
TGHGL	$\overline{PROG}$ High to $\overline{PROG}$ Low	10		μs



EPROM Programming and Verification Waveforms

80C51BH/80C51BH-1/80C51BH-2
CHMOS SINGLE-CHIP 8-BIT MICROCOMPUTER WITH FACTORY MASK-PROGRAMMABLE ROM
80C31BH/80C31BH-1/80C31BH-2
CHMOS SINGLE-CHIP 8-BIT CONTROL-ORIENTED CPU WITH RAM AND I/O

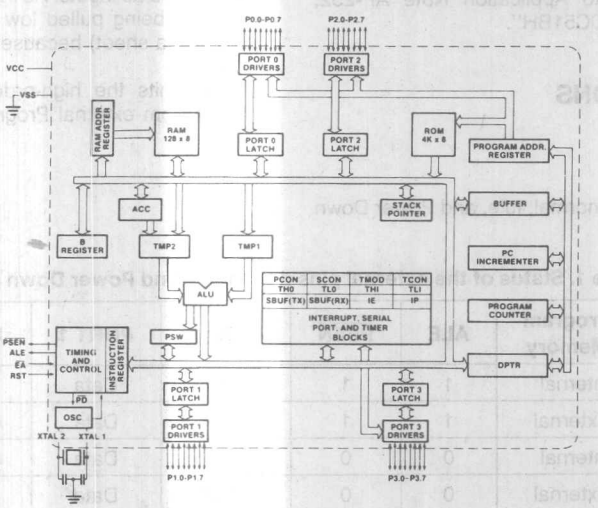
80C51BH/80C31BH—3.5 to 12 MHz, V_{CC} = 5V ±20%
80C51BH-1/80C31BH-1—3.5 to 16 MHz, V_{CC} = 5V ±20%
80C51BH-2/80C31BH-2—0.5 to 12 MHz, V_{CC} = 5V ±20%

- **Power Control Modes**
- **128 x 8-Bit RAM**
- **32 Programmable I/O Lines**
- **Two 16-Bit Timer/Counters**
- **64K Program Memory Space**
- **High Performance CHMOS Process**
- **Boolean Processor**
- **5 Interrupt Sources**
- **Programmable Serial Port**
- **64K Data Memory Space**

The MCS®-51 CHMOS products are fabricated on Intel's CHMOS III process and are functionally compatible with the standard MCS-51 HMOS and EPROM products. CHMOS III is a technology which combines the high speed and density characteristics of HMOS with the low power attributes of CHMOS. This combination expands the effectiveness of the powerful MCS-51 architecture and instruction set.

Like the MCS-51 HMOS versions, the MCS-51 CHMOS products have the following features: 4K byte of ROM (80C51BH/80C51BH-1/80C51BH-2 only); 128 bytes of RAM; 32 I/O lines; two 16-bit timer/counters; a five-source two-level interrupt structure; a full duplex serial port; and on-chip oscillator and clock circuitry. In addition, the MCS-51 CHMOS products have two software selectable modes of reduced activity for further power reduction—Idle and Power Down.

The Idle mode freezes the CPU while allowing the RAM, timer/counters serial port and interrupt system to continue functioning. The Power Down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.



270064–1

Figure 1. Block Diagram

September 1987
Order Number: 270064-005

IDLE MODE

In the Idle mode, the CPU puts itself to sleep while all the on chip peripherals stay active. The instruction that invokes the Idle mode is the last instruction executed in the normal operating mode before Idle mode is activated. The content of CPU, the on chip RAM, and all the Special Function Registers remain intact during this mode. The Idle mode can be terminated either by any enabled interrupt, at which time the process is picked up at the interrupt service routine and continued, or by a hardware reset which starts the processor the same as a power on reset.

POWER DOWN MODE

In the Power Down mode the oscillator is stopped, and the instruction that invokes Power Down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the Power Down mode is terminated.

The only exit from Power Down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

The control bits for the reduced power modes are in the Special Function Register PCON.

NOTE:
For more detailed information on these reduced power modes refer to Application Note AP-252, "Designing with the 80C51BH".

PIN DESCRIPTIONS

V_{CC}

Supply voltage during normal, Idle, and Power Down operations.

V_{SS}

Circuit ground.

Port 0

Port 0 is an 8-bit open drain bi-directional I/O port. Port 0 pins that have 1's written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s. Port 0 also outputs the code bytes during program verification in the 80C51BH. External pullups are required during program verification.

Port 1

Port 1 is an 8-bit bidirectional I/O port with internal pullups. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL, on the data sheet) because of the internal pullups.

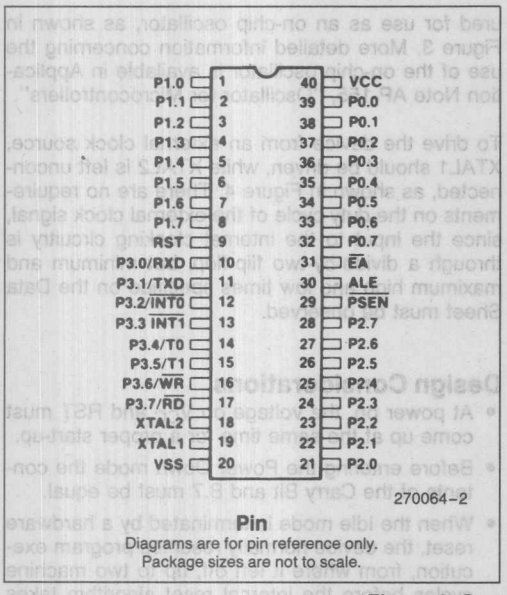
Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pullups. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (IIL, on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during

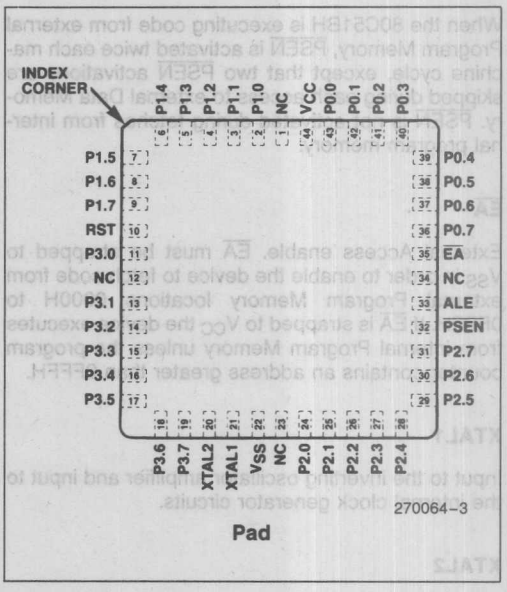
Table 1. Status of the external pins during Idle and Power Down modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data



Pin

Diagrams are for pin reference only.
Package sizes are not to scale.



Pad

Figure 2. Connection Diagrams

accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pullups. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (IIL, on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. An internal diffused resistor to V_{SS} permits Power-On reset using only an external capacitor to V_{CC}.

ALE

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

$\overline{PSEN}$

Program Store Enable is the read strobe to external Program Memory.

When the 80C51BH is executing code from external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external Data Memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.

$\overline{EA}$

External Access enable. $\overline{EA}$ must be strapped to V_{SS} in order to enable the device to fetch code from external Program Memory locations 0000H to 0FFFH. If $\overline{EA}$ is strapped to V_{CC} the device executes from internal Program Memory unless the program counter contains an address greater than 0FFFH.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock generator circuits.

XTAL2

Output from the inverting oscillator amplifier.

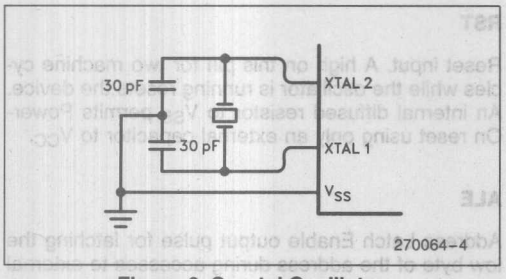


Figure 3. Crystal Oscillator

Oscillator Characteristics

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillator for Microcontrollers".

To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left unconnected, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

Design Considerations

- At power on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.
- Before entering the Power Down mode the contents of the Carry Bit and B.7 must be equal.
- When the Idle mode is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

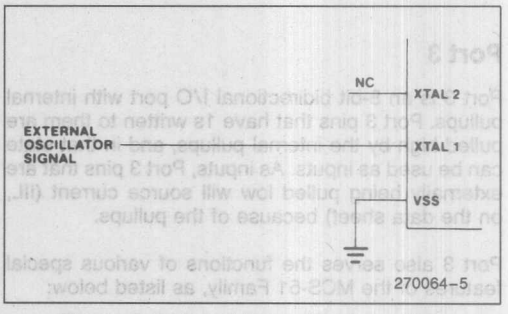


Figure 4. External Drive Configuration

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on any Pin to V_{SS} . . . . . −0.5V to V_{CC} +0.5V
Voltage on V_{CC} to V_{SS} . . . . . −0.5V to 6.5V
Power Dissipation 1.0W*

*This value is based on the maximum allowable die temperature and the thermal resistance of the package.

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS (T_A = 0°C to 70°C; V_{CC} = 5V ±20%; V_{SS} = 0V)

Symbol	Parameter	Min	Typ(3)	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage (Except $\overline{EA}$)	−0.5		0.2 V_{CC} −0.1	V	
V_{IL1}	Input Low Voltage ($\overline{EA}$)	−0.5		0.2 V_{CC} − 0.3	V	
V_{IH}	Input High Voltage (Except XTAL1, RST)	0.2 V_{CC} + 0.9		V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage (XTAL1, RST)	0.7 V_{CC}		V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage (Ports 1, 2, 3)			0.45	V	I_{OL} = 1.6 mA (1)
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$)			0.45	V	I_{OL} = 3.2 mA (1)
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, $\overline{PSEN}$)	2.4			V	I_{OH} = −60 μA V_{CC} = 5V ±10%
		0.75 V_{CC}			V	I_{OH} = −25 μA
		0.9 V_{CC}			V	I_{OH} = −10 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4			V	I_{OH} = −800 μA V_{CC} = 5V ±10%
		0.75 V_{CC}			V	I_{OH} = −300 μA
		0.9 V_{CC}			V	I_{OH} = −80 μA (2)
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3)			−50	μA	V_{IN} = 0.45V
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, 3)			−650	μA	V_{IN} = 2V
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)			±10	μA	0.45 < V_{IN} < V_{CC}
RRST	Reset Pulldown Resistor	50		150	KΩ	
CIO	Pin Capacitance			10	pF	Test Freq = 1 MHz, T_A = 25°C
I_{CC}	Power Supply Current:					
	Active Mode, 12 MHz (4)		11	20	mA	
	Idle Mode, 12 MHz (4)		1.7	5	mA	(5)
	Power Down Mode		5	50	μA	

NOTES:
1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OLS} of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
2. Capacitive loading on Ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the 0.9 V_{CC} specification when the address bits are stabilizing.
3. "Typicals" are based on a limited number of samples taken from early manufacturing lots and are not guaranteed. The values listed are at room temperature, 5V.
4. ICCMAX at other frequencies is given by
   Active Mode: ICCMAX = 1.47 × FREQ + 2.35
   Idle Mode: ICCMAX = 0.33 × FREQ + 1.05
where FREQ is the external oscillator frequency in MHz. ICCMAX is given in mA. See Figure 5.
5. See Figures 6 through 9 for I_{CC} test conditions.

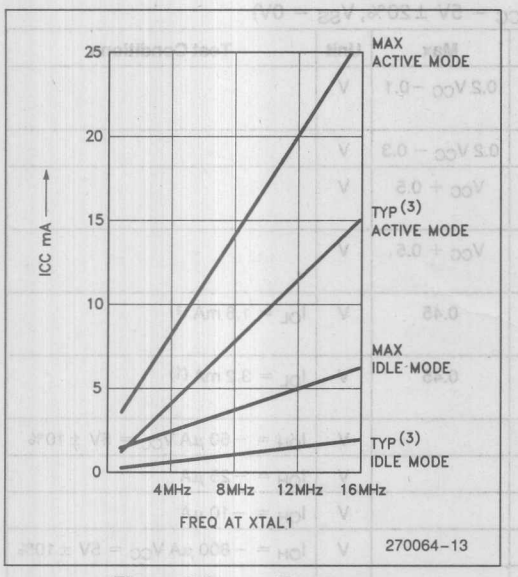


Figure 5. I_{CC} vs. Frequency. Valid only within frequency specifications of the device under test.

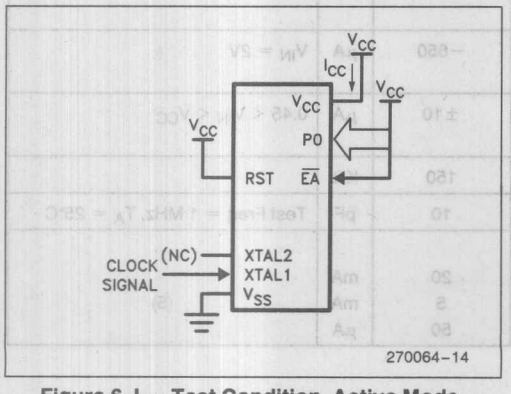


Figure 6. I_{CC} Test Condition, Active Mode. All other pins are disconnected.

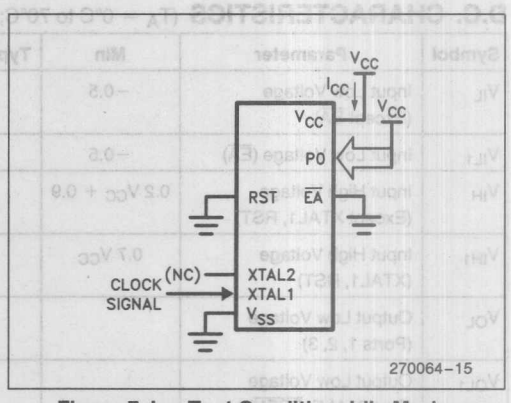


Figure 7. I_{CC} Test Condition, Idle Mode. All other pins are disconnected.

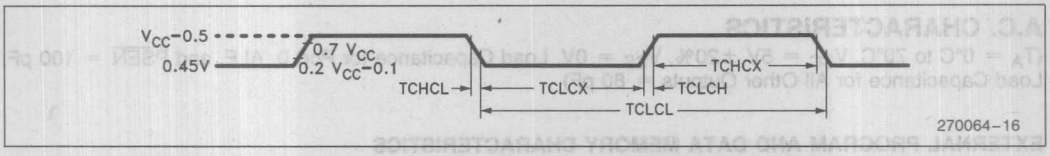


Figure 8. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes. TCLCH = TCHCL = 5 ns.

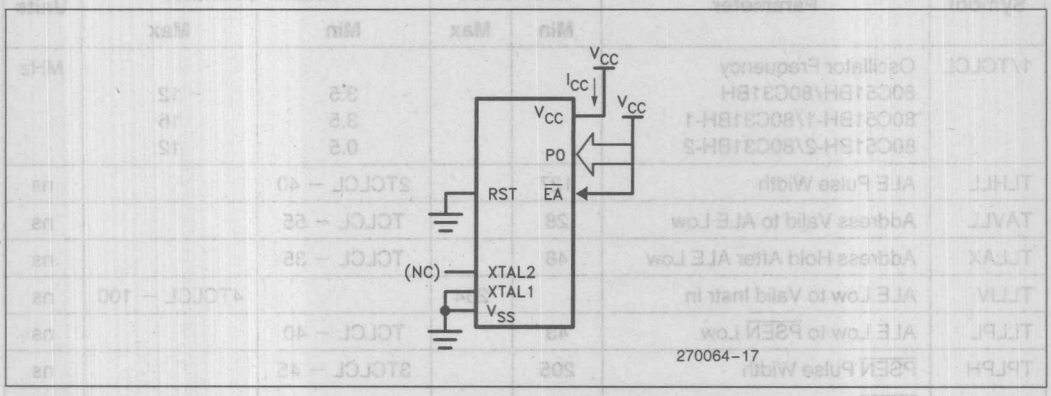


Figure 9. I_{CC} Test Condition, Power Down Mode. All other pins are disconnected. V_{CC} = 2V to 6V.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

A: Address.
C: Clock.
D: Input data.
H: Logic level HIGH.
I: Instruction (program memory contents).
L: Logic level LOW, or ALE.
P: $\overline{PSEN}$.
Q: Output data.
R: $\overline{RD}$ signal.
T: Time.
V: Valid.
W: $\overline{WR}$ signal.
X: No longer a valid logic level.
Z: Float.

EXAMPLE:

TAVLL = Time for Address Valid to ALE Low.
TLLPL = Time for ALE Low to $\overline{PSEN}$ Low.

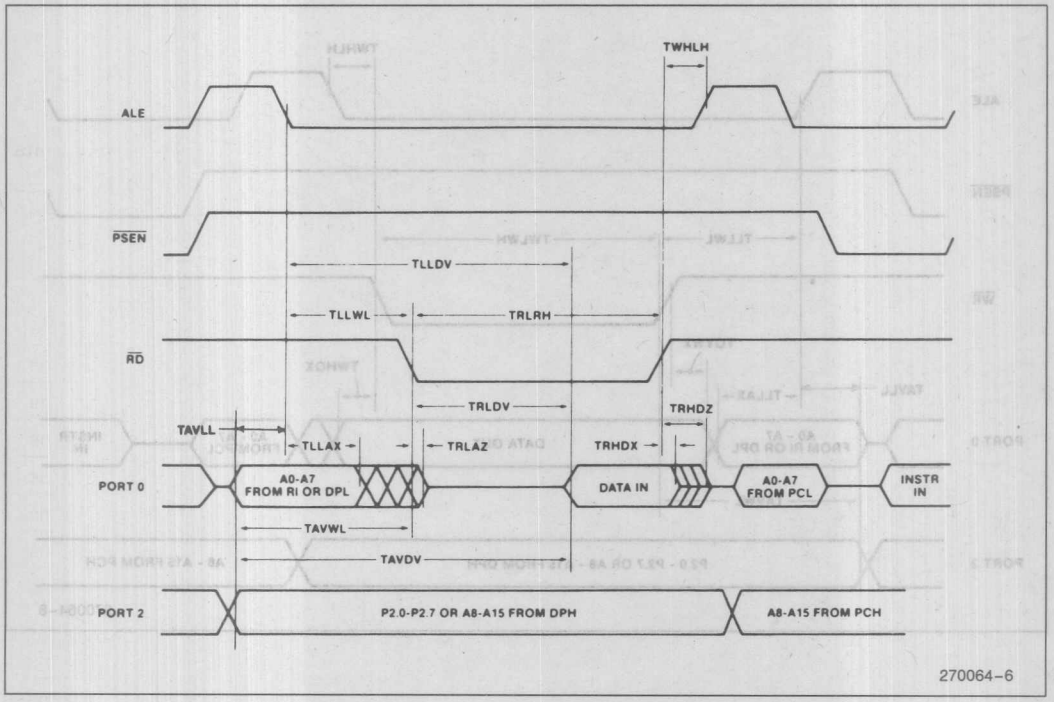
A.C. CHARACTERISTICS

(T_A = 0°C to 70°C, V_{CC} = 5V ±20%, V_{SS} = 0V, Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF, Load Capacitance for All Other Outputs = 80 pF)

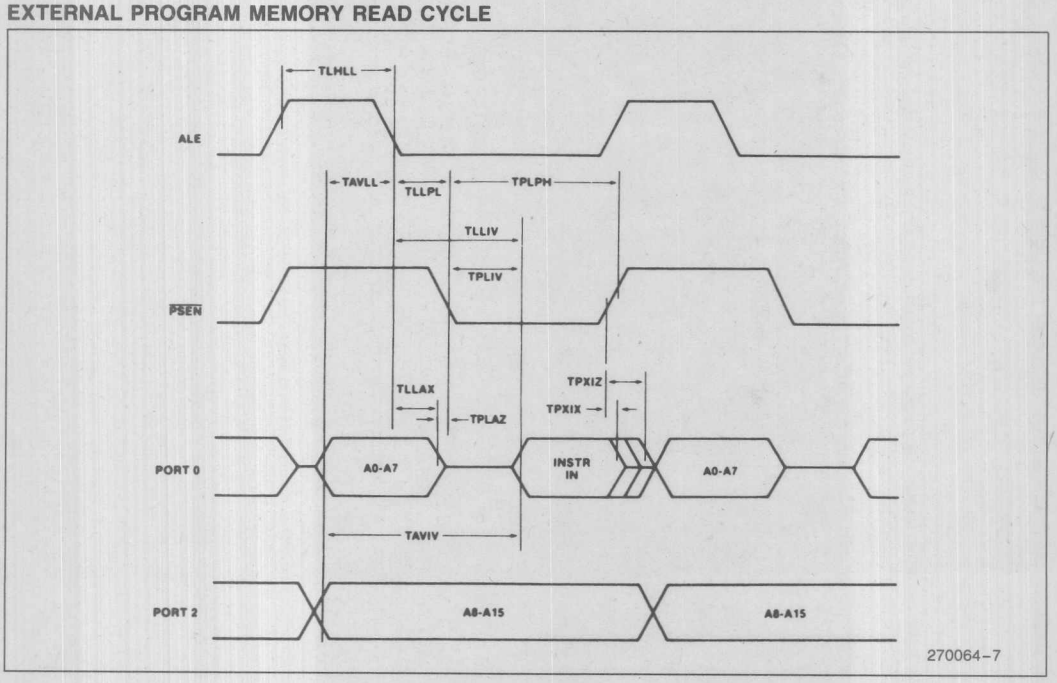
EXTERNAL PROGRAM AND DATA MEMORY CHARACTERISTICS

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency 80C51BH/80C31BH 80C51BH-1/80C31BH-1 80C51BH-2/80C31BH-2			 3.5 3.5 0.5	 12 16 12	MHz
TLHLL	ALE Pulse Width	127		2TCLCL − 40		ns
TAVLL	Address Valid to ALE Low	28		TCLCL − 55		ns
TLLAX	Address Hold After ALE Low	48		TCLCL − 35		ns
TLLIV	ALE Low to Valid Instr In		234		4TCLCL − 100	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	43		TCLCL − 40		ns
TPLPH	$\overline{PSEN}$ Pulse Width	205		3TCLCL − 45		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr In		145		3TCLCL − 105	ns
TPXIX	Input Instr Hold After $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float After $\overline{PSEN}$		59		TCLCL − 25	ns
TAVIV	Address to Valid Instr In		312		5TCLCL − 105	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		10		10	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL − 100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL − 100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		252		5TCLCL − 165	ns
TRHDX	Data Hold After $\overline{RD}$	0		0		ns
TRHDZ	Data Float After $\overline{RD}$		97		2TCLCL − 70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL − 150	ns
TAVDV	Address to Valid Data In		585		9TCLCL − 165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL − 50	3TCLCL + 50	ns
TAVWL	Address Valid to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL − 130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition	23		TCLCL − 60		ns
TWHQX	Data Hold After $\overline{WR}$	33		TCLCL − 50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	TCLCL − 40	TCLCL + 40	ns

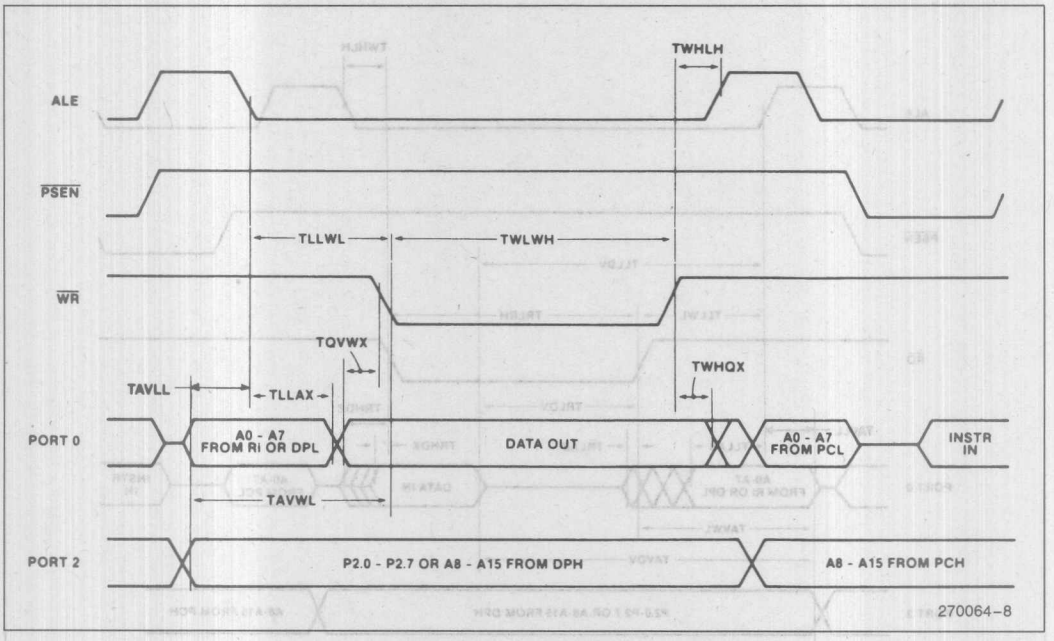
EXTERNAL DATA MEMORY READ CYCLE

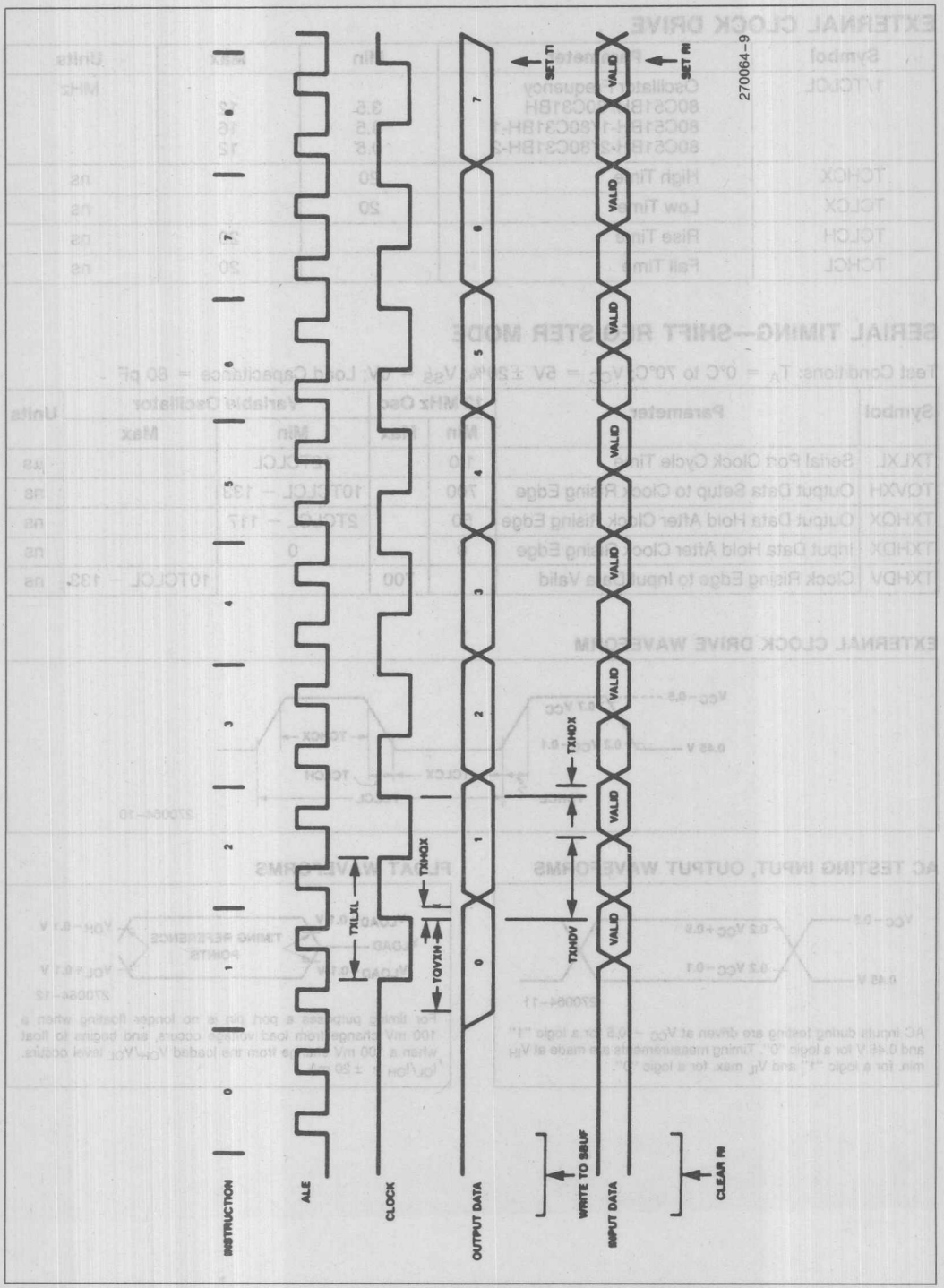


EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE





Shift Register Mode Timing Waveforms

EXTERNAL CLOCK DRIVE

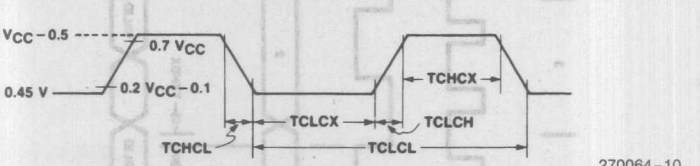
Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency 80C51BH/80C31BH 80C51BH-1/80C31BH-1 80C51BH-2/80C31BH-2	 3.5 3.5 0.5	 12 16 12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

SERIAL TIMING—SHIFT REGISTER MODE

Test Conditions: T_A = 0°C to 70°C; V_{CC} = 5V ±20%; V_{SS} = 0V; Load Capacitance = 80 pF

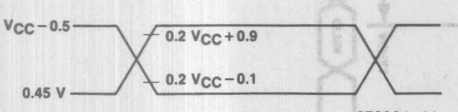
Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL − 133		ns
TXHQX	Output Data Hold After Clock Rising Edge	50		2TCLCL − 117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL − 133	ns

EXTERNAL CLOCK DRIVE WAVEFORM



270064-10

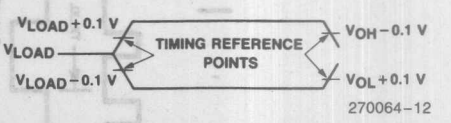
AC TESTING INPUT, OUTPUT WAVEFORMS



270064-11

AC Inputs during testing are driven at V_{CC} − 0.5 for a logic "1" and 0.45 V for a logic "0". Timing measurements are made at V_{IH} min. for a logic "1" and V_{IL} max. for a logic "0".

FLOAT WAVEFORMS



270064-12

For timing purposes a port pin is no longer floating when a 100 mV change from load voltage occurs, and begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OL}/I_{OH} \geq \pm 20$ mA.

80C31BH/80C51BH

EXPRESS

- **Extended Temperature Range**
- **3.5 to 12 MHz V_{CC} = 5V ±20%**
- **Burn-In**

The Intel EXPRESS system offers enhancements to the operational specifications of the MCS®-51 family of microcontrollers. These EXPRESS products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

The EXPRESS program includes the commercial standard temperature range with burn-in and an extended temperature range with or without burn-in.

With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to 70°C. With the extended temperature range option, operational characteristics are guaranteed over the range of −40°C to +85°C.

The optional burn-in is dynamic for a minimum time of 160 hours at 125°C with V_{CC} = 6.9V ±0.25V, following guidelines in MIL-STD-883, Method 1015.

Package types and EXPRESS versions are identified by a one- or two-letter prefix to the part number. The prefixes are listed in Table 1.

For the extended temperature range option, this data sheet specifies the parameters which deviate from their commercial temperature range limits. The commercial temperature range data sheets are applicable for all parameters not listed here.

September 1987
Order Number: 270218-002

Electrical Deviations from Commercial Specifications for Extended Temperature Range

D.C. and A.C. parameters not included here are the same as in the commercial temperature range data sheets.

D.C. CHARACTERISTICS $T_A = -40°C$ to $+85°C$; $V_{CC} = 5V \pm 20\%$; $V_{SS} = 0V$

Symbol	Parameter	Limits		Unit	Test Conditions
		Min	Max		
V_{IL}	Input Low Voltage (Except EA)	-0.5	$0.2V_{CC} - 0.15$	V	
V_{IL1}	EA	$-0.5V$	$0.2V_{CC} - 0.35$	V	
V_{IH}	Input High Voltage (Except XTAL1, RST)	$0.2V_{CC} + 1$	$V_{CC} + 0.5$	V	
V_{IH1}	Input High Voltage to XTAL1, RST	$0.7V_{CC} + 0.1$	$V_{CC} + 0.5$	V	
I_{IL}	Logical 0 Input Current (Port 1, 2, 3)		-75	μA	$V_{in} = 0.45V$
I_{TL}	Logical 1 to 0 transition Current (Ports 1, 2, 3)		-750	μA	$V_{in} = 2.0V$

Table 1. Prefix Identification

Prefix	Package Type	Temperature Range	Burn-In
P	Plastic	Commercial	No
D	Cerdip	Commercial	No
N	PLCC	Commercial	No
TP	Plastic	Extended	No
TD	Cerdip	Extended	No
TN	PLCC	Extended	No
QP	Plastic	Commercial	Yes
QD	Cerdip	Commercial	Yes
QN	PLCC	Commercial	Yes
LP	Plastic	Extended	Yes
LD	Cerdip	Extended	Yes
LN	PLCC	Extended	Yes

NOTE:
- Commercial temperature range is 0°C to 70°C. Extended temperature range is $-40°C$ to $+85°C$.
- Burn-in is dynamic for a minimum time of 160 hours at 125°C, $V_{CC} = 6.9V \pm 0.25V$, following guidelines in MIL-STD-883 Method 1015 (Test Condition D).

Examples:

P80C31BH indicates 80C31BH in a plastic package and specified for commercial temperature range, without burn-in.
LD80C51BH indicates 80C51BH in a cerdip package and specified for extended temperature range with burn-in.

87C51/87C51-1/87C51-2
CHMOS SINGLE-CHIP 8-BIT MICROCONTROLLER WITH 4K BYTES OF EPROM PROGRAM MEMORY

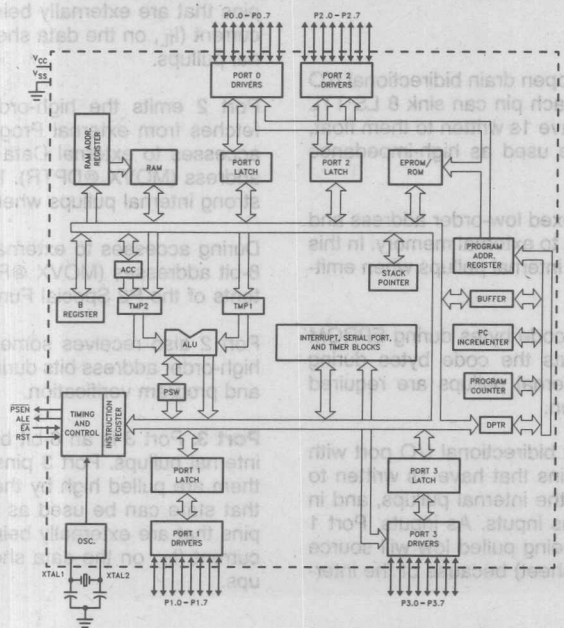
87C51—3.5 to 12 MHz, V_{CC} = 5V ± 10%
87C51-1—3.5 to 16 MHz, V_{CC} = 5V ± 10%
87C51-2—0.5 to 12 MHz, V_{CC} = 5V ± 10%

- High Performance CHMOS EPROM
- Quick-Pulse Programming™ Algorithm
- 2-Level Program Memory Lock
- Boolean Processor
- 128-Byte Data RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- 5 Interrupt Sources
- Programmable Serial Channel
- TTL- and CMOS-Compatible Logic Levels
- 64K External Program Memory Space
- 64K External Data Memory Space
- IDLE and POWER DOWN Modes
- ONCE™ Mode Facilitates System Testing
- LCC, PLCC, and DIP Packaging Available

The 87C51 is the EPROM version of the 80C51BH. It is fabricated on Intel's CHMOS II-E process. It contains 4K bytes of on-chip Program memory that can be electrically programmed, and can be erased by exposure to ultraviolet light.

The 87C51 EPROM array uses a modified Quick-Pulse programming algorithm, by which the entire 4K-byte array can be programmed in about 12 seconds.

The extremely low operating power, along with the two reduced power modes, Idle and Power Down, make this part very suitable for low power applications. The Idle mode freezes the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power Down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.



270147–1

Figure 1. MCS®-51 Architectural Block Diagram

October 1987
Order Number: 270147-004

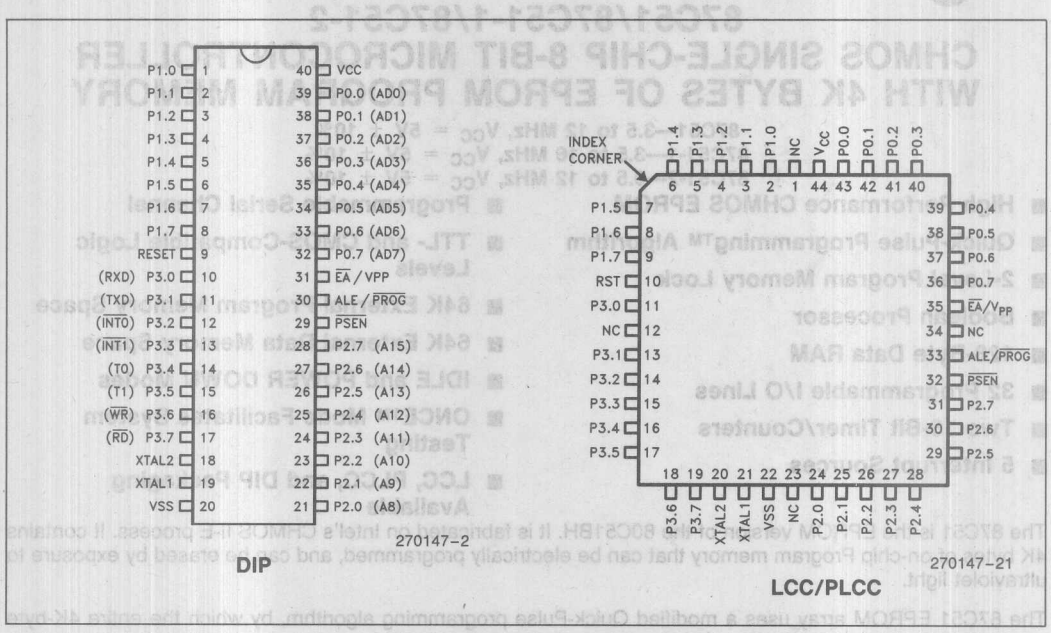


Figure 2. Pin Connections

PIN DESCRIPTION

V_{CC}: Supply voltage during normal, Idle, and Power Down operations.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external memory. In this application it uses strong internal pullups when emitting 1s.

Port 0 also receives the code bytes during EPROM programming, and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during EPROM programming and program verification.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program memory and during accesses to external Data Memory that use 16-bit address (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s.

During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives some control signals and the high-order address bits during EPROM programming and program verification.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Pin	Name	Alternate Function
P3.0	RXD	Serial input line
P3.1	TXD	Serial output line
P3.2	$\overline{INT0}$	External Interrupt 0
P3.3	$\overline{INT1}$	External Interrupt 1
P3.4	T0	Timer 0 external input
P3.5	T1	Timer 1 external input
P3.6	$\overline{WR}$	External Data Memory Write strobe
P3.7	$\overline{RD}$	External Data Memory Read strobe

Port 3 also receives some control signals for EPROM programming and program verification.

RST: Reset input. A logic high on this pin for two machine cycles while the oscillator is running resets the device. An internal pulldown resistor permits a power-on reset to be generated using only an external capacitor to V_{CC}.

ALE/$\overline{PROG}$: Address Latch Enable output signal for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

$\overline{PSEN}$: Program Store Enable is the Read strobe to External Program Memory. When the 87C51 is executing from Internal Program Memory, $\overline{PSEN}$ is inactive (high). When the device is executing code from External Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to External Data Memory.

$\overline{EA}$/V_{PP}: External Access enable. $\overline{EA}$ must be strapped to V_{SS} in order to enable the 87C51 to fetch code from External Program Memory locations 0000H to 0FFFH. Note, however, that if either of the Lock Bits is programmed, the logic level at $\overline{EA}$ is internally latched during reset.

$\overline{EA}$ must be strapped to V_{CC} for internal program execution.

This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.

XTAL1: Input to the inverting oscillator amplifier and input to the internal clock generating circuits.

XTAL2: Output from the inverting oscillator amplifier.

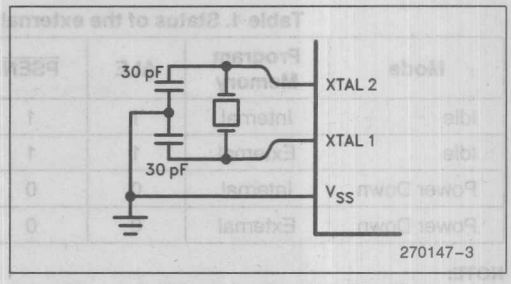


Figure 3. Using the On-Chip Oscillator

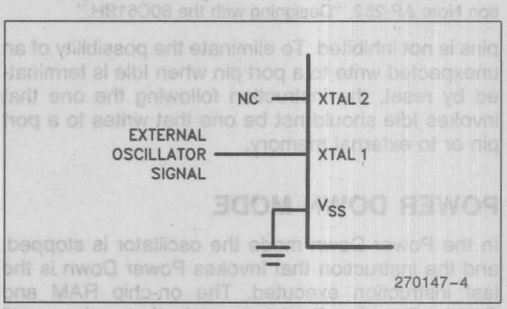


Figure 4. External Clock Drive

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3.

To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left unconnected, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

IDLE MODE

In Idle Mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the Special Functions Registers remain unchanged during this mode. The Idle Mode can be terminated by any enabled interrupt or by a hardware reset.

It should be noted that when Idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port

Mode	Memory	ALE	PSEN	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data

NOTE:
For more detailed information on the reduced power modes refer to current Embedded Controller Handbook, and Application Note AP-252, "Designing with the 80C51BH."

pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

POWER DOWN MODE

In the Power Down mode the oscillator is stopped, and the instruction that invokes Power Down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the Power Down mode is terminated.

The only exit from Power Down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

DESIGN CONSIDERATIONS

Exposure to light when the device is in operation may cause logic errors. For this reason, it is suggested that an opaque label be placed over the window when the die is exposed to ambient light.

If using the 87C51 to prototype for the 80C51BH, consult the Design Considerations section of the 80C51BH data sheet.

PROGRAM MEMORY LOCK

The 87C51 contains two program memory lock schemes: Encrypted Verify and Lock Bits.

Encrypted Verify: The 87C51 implements a 32-byte EPROM array that can be programmed by the customer, and which can then be used to encrypt the program code bytes during EPROM verification. The EPROM verification procedure is performed as usual, except that each code byte comes out logically X-NORed with one of the 32 key bytes. The key bytes are gone through in sequence. Therefore, to read the ROM code, one has to know the 32 key bytes in their proper sequence.

Lock Bits: Also on the chip are two Lock Bits which can be left unprogrammed (U) or can be programmed (P) to obtain the following additional features:

Bit 1	Bit 2	Additional Features
U	U	none
P	U	• Externally fetched code can not access internal Program Memory. • Further programming disabled.
U	P	(Reserved for Future definition.)
P	P	• Externally fetched code can not access internal Program Memory. • Further programming disabled. • Program verification is disabled.

When Lock Bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

ONCE™ MODE

The ONCE ("on-circuit emulation") mode facilitates testing and debugging of systems using the 87C51 without the 87C51 having to be removed from the circuit. The ONCE mode is invoked by:

1. Pull ALE low while the device is in reset and $\overline{PSEN}$ is high;
2. Hold ALE low as RST is deactivated.

While the device is in ONCE mode, the Port 0 pins go into a float state, and the other port pins and ALE and $\overline{PSEN}$ are weakly pulled high. The oscillator circuit remains active. While the 87C51 is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored when a normal reset is applied.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on $\overline{EA}/V_{PP}$ Pin to V_{SS} . . . . . 0V to +13.0V
Voltage on Any Other Pin to V_{SS} . . −0.5V to +6.5V
Power Dissipation 1.5W
(Based on package heat transfer limitations, not device power consumption).

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS: (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V)

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage (Except $\overline{EA}$)	−0.5		$.2V_{CC}-.1$	V	
V_{IL1}	Input Low Voltage to $\overline{EA}$	0		$.2V_{CC}-.3$	V	
V_{IH}	Input High Voltage (Except XTAL1, RST)	$.2V_{CC}+.9$		$V_{CC}+.5$	V	
V_{IH1}	Input High Voltage (XTAL1, RST)	$0.7V_{CC}$		$V_{CC}+.5$	V	
V_{OL}	Output Low Voltage (Ports 1, 2, 3)			0.45	V	I_{OL} = 1.6 mA (2)
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$)			0.45	V	I_{OL} = 3.2 mA (2)
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, $\overline{PSEN}$)	2.4			V	I_{OH} = −60 μA
		$.75V_{CC}$			V	I_{OH} = −25 μA
		$.9V_{CC}$			V	I_{OH} = −10 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4			V	I_{OH} = −800 μA
		.75 V_{CC}			V	I_{OH} = −300 μA
		$.9V_{CC}$			V	I_{OH} = −80 μA (3)
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3)			−50	μA	V_{IN} = 0.45 V
I_{TL}	Logical 1-to-0 transition current (Ports 1, 2, 3)			−650	μA	(4)
I_{LI}	Input Leakage Current (Port 0)			±10	μA	$V_{IN} = V_{IL}$ or V_{IH}
I_{CC}	Power Supply Current:					
	Active Mode @ 12 MHz (5)		11.5	25	mA	
	Idle Mode @ 12 MHz (5)		1.3	4	mA	(6)
	Power Down Mode		3	50	μA	
RRST	Internal Reset Pulldown Resistor	50		300	kΩ	
C_{IO}	Pin Capacitance			10	pF	

NOTES:
1. "Typicals" are based on a limited number of samples taken from early manufacturing lots and are not guaranteed. The values listed are at room temp, 5V.
2. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
3. Capacitive loading on Ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. Pins of Ports 1, 2, and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CC}MAX at other frequencies is given by:

Active Mode: $I_{CC}MAX = 0.94 \times FREQ + 13.71$
Idle Mode: $I_{CC}MAX = 0.14 \times FREQ + 2.31$

where FREQ is the external oscillator frequency in MHz. I_{CC}MAX is given in mA. See Figure 5.
6. See Figures 6 through 9 for I_{CC} test conditions.

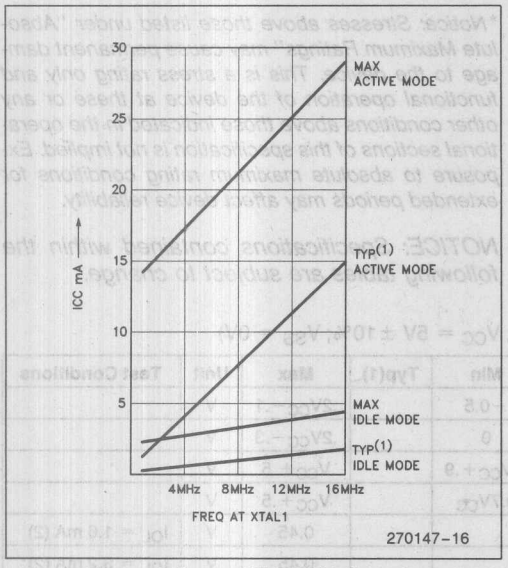


Figure 5. I_{CC} vs. FREQ. Valid only within frequency specifications of the device under test.

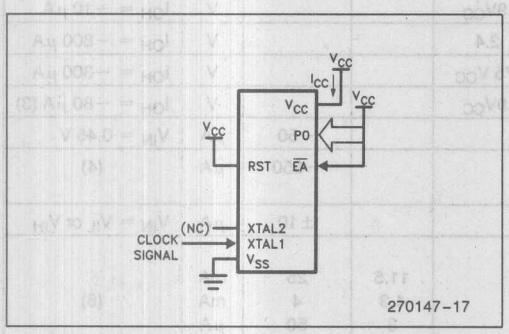


Figure 6. I_{CC} Test Condition, Active Mode. All other pins are disconnected.

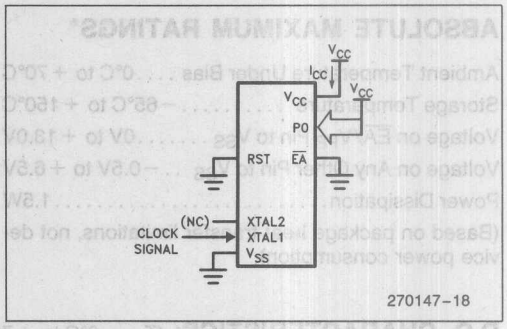


Figure 7. I_{CC} Test Condition, Idle Mode. All other pins are disconnected.

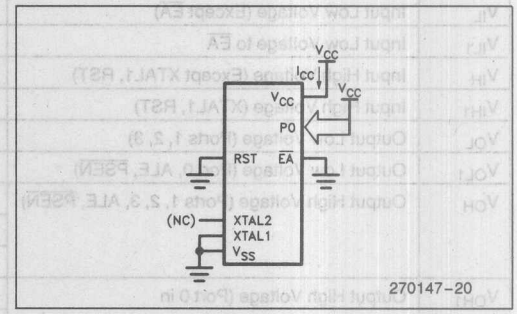


Figure 9. I_{CC} Test Condition, Power Down Mode. All other pins are disconnected. V_{CC} = 2V to 5.5V.

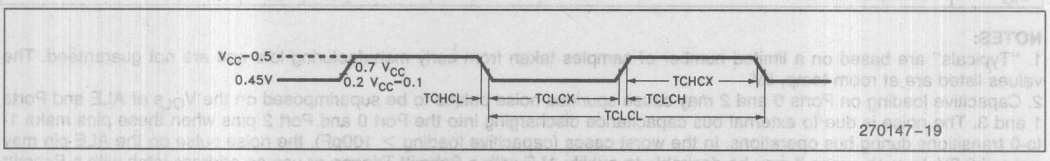


Figure 8. Clock Signal Waveform for I_{CC} tests in Active and Idle Modes. TCLCH = TCHCL = 5 ns.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

A:Address.
C:Clock.
D:Input data.
H:Logic level HIGH.
I:Instruction (program memory contents).
L:Logic level LOW, or ALE.
P:$\overline{PSEN}$.
Q:Output data.
R:$\overline{RD}$ signal.
T:Time.
V:Valid.
W:$\overline{WR}$ signal.
X:No longer a valid logic level.
Z:Float.

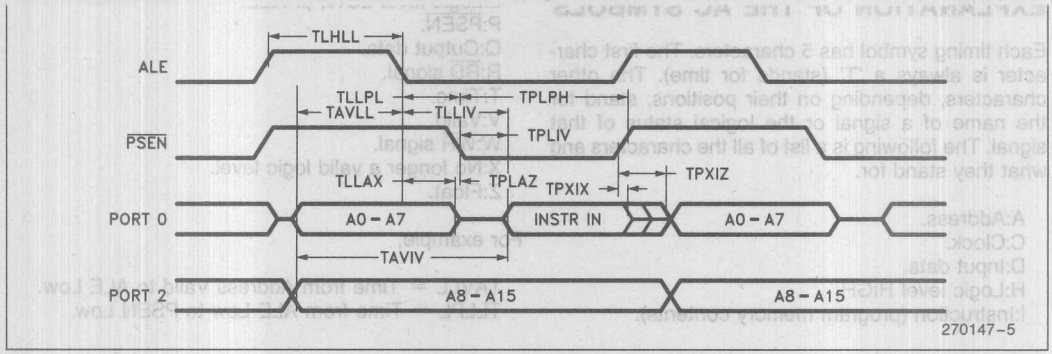
For example,

TAVLL = Time from Address Valid to ALE Low.
TLLPL = Time from ALE Low to PSEN Low.

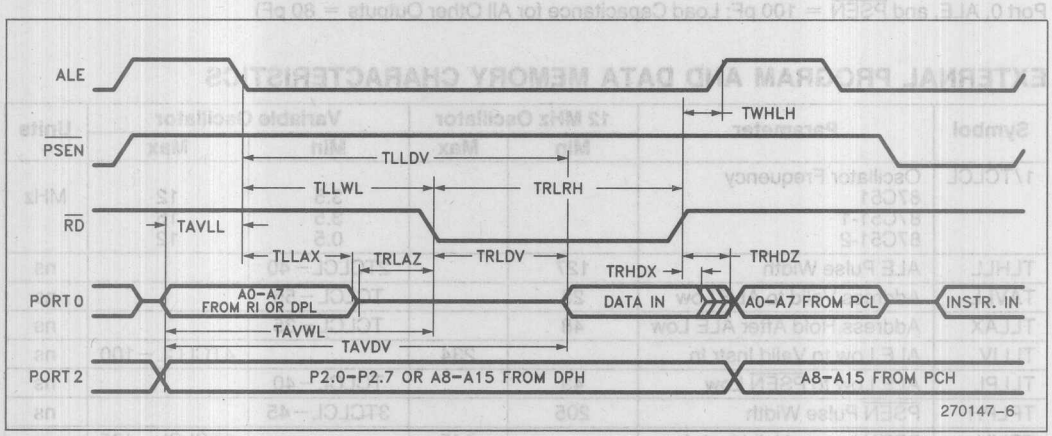
A.C. CHARACTERISTICS: (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF; Load Capacitance for All Other Outputs = 80 pF)

EXTERNAL PROGRAM AND DATA MEMORY CHARACTERISTICS

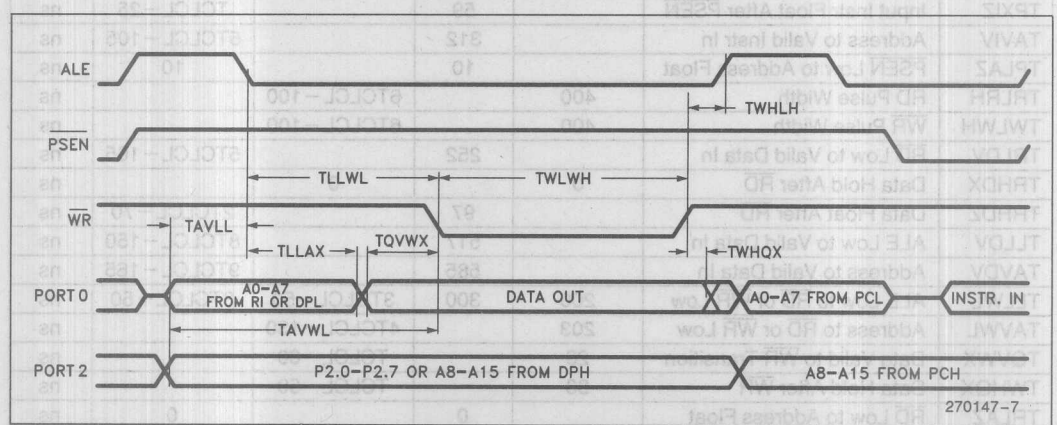
Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency 87C51 87C51-1 87C51-2			 3.5 3.5 0.5	 12 16 12	MHz
TLHLL	ALE Pulse Width	127		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	28		TCLCL−55		ns
TLLAX	Address Hold After ALE Low	48		TCLCL−35		ns
TLLIV	ALE Low to Valid Instr In		234		4TCLCL−100	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	43		TCLCL−40		ns
TPLPH	$\overline{PSEN}$ Pulse Width	205		3TCLCL−45		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr In		145		3TCLCL−105	ns
TPXIX	Input Instr Hold After $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float After $\overline{PSEN}$		59		TCLCL−25	ns
TAVIV	Address to Valid Instr In		312		5TCLCL−105	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		10		10	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL−100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL−100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		252		5TCLCL−165	ns
TRHDX	Data Hold After $\overline{RD}$	0		0		ns
TRHDZ	Data Float After $\overline{RD}$		97		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL−150	ns
TAVDV	Address to Valid Data In		585		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition	23		TCLCL−60		ns
TWHQX	Data Hold After $\overline{WR}$	33		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	TCLCL−40	TCLCL+40	ns



External Program Memory Read Cycle



External Data Memory Read Cycle

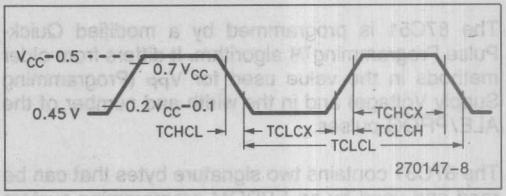


External Data Memory Write Cycle

EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency			
	87C51	3.5	12	MHz
	87C51-1	3.5	16	
	87C51-2	0.5	12	
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

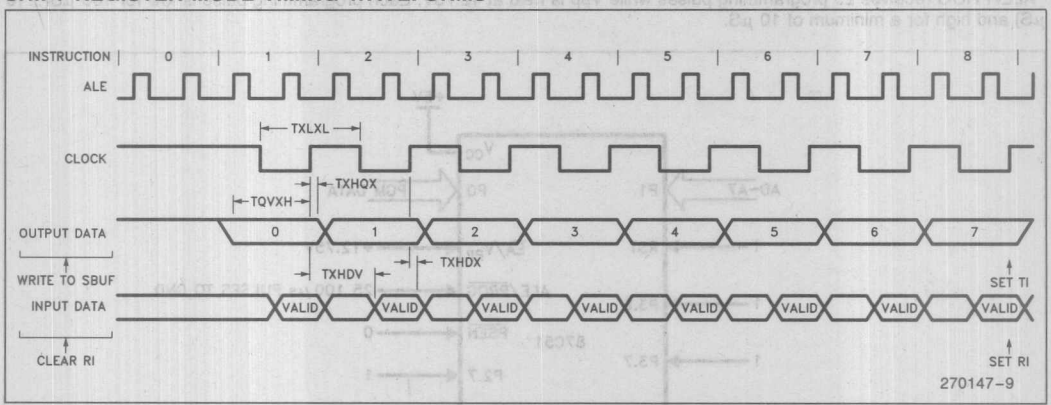
EXTERNAL CLOCK DRIVE WAVEFORM



SERIAL PORT TIMING—SHIFT REGISTER MODE

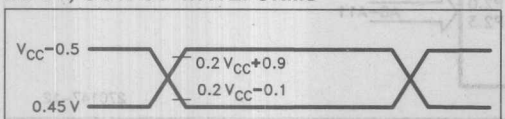
Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL−133		ns
TXHQX	Output Data Hold After Clock Rising Edge	50		2TCLCL−117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL−133	ns

SHIFT REGISTER MODE TIMING WAVEFORMS



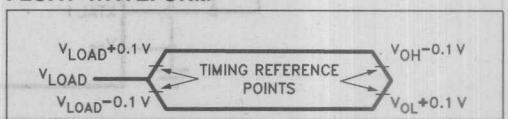
A.C. TESTING:

INPUT, OUTPUT WAVEFORMS



AC inputs during testing are driven at V_{CC} − 0.5 for a Logic "1" and 0.45V for a Logic "0." Timing measurements are made at V_{IH} min for a Logic "1" and V_{IL} max for a Logic "0".

FLOAT WAVEFORM



For timing purposes a port pin is no longer floating when a 100 mV change from load voltage occurs, and begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs. I_{OL}/I_{OH} ≥ ±20 mA.

EPROM CHARACTERISTICS

The 87C51 is programmed by a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (Programming Supply Voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C51 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C51 manufactured by Intel.

Table 2 shows the logic levels for reading the signature byte, and for programming the Program Memory, the Encryption Table, and the Lock Bits. The circuit configuration and waveforms for Quick-Pulse Programming™ are shown in Figures 10 and 11. Figure 12 shows the circuit configuration for normal Program Memory verification.

Table 2. EPROM Programming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read Signature	1	0	1	1	0	0	0	0
Program Code Data	1	0	0*	V_{PP}	1	0	1	1
Verify Code Data	1	0	1	1	0	0	1	1
Pgm Encryption Table	1	0	0*	V_{PP}	1	0	1	0
Pgm Lock Bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm Lock Bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:
"1" = Valid high for that pin
"0" = Valid low for that pin
V_{PP} = **12.75V ± 0.25V**
V_{CC} = **5V ± 10%** during programming and verification
*ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100 μS(±10 μS) and high for a minimum of 10 μS.

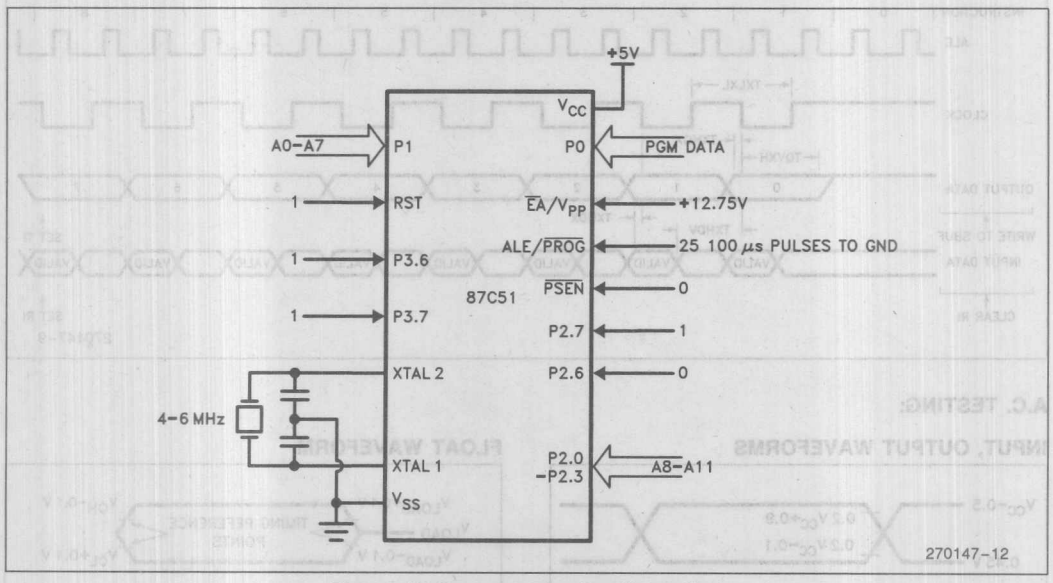


Figure 10. Programming Configuration

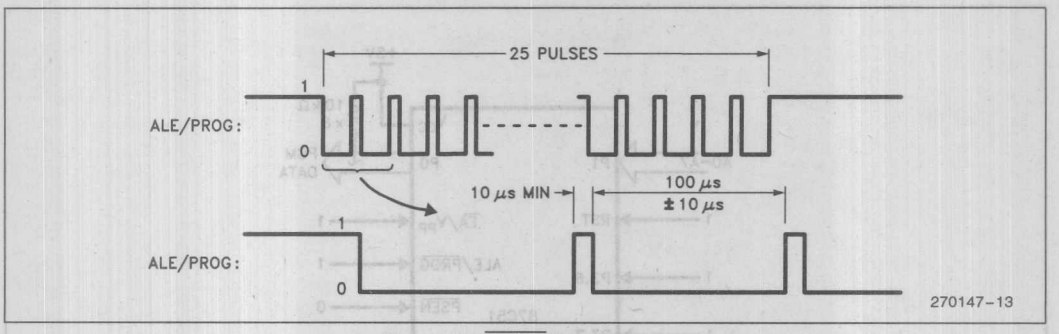


Figure 11. $\overline{\text{PROG}}$ Waveforms

Quick-Pulse Programming™

The setup for Microcontroller Quick-Pulse Programming™ is shown in Figure 10. Note that the 87C51 is running with a 4 to 6 MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to Ports 1 and 2, as shown in Figure 10. The code byte to be programmed into that location is applied to Port 0. RST, $\overline{\text{PSEN}}$, and pins of Ports 2 and 3 specified in Table 2 are held at the "Program Code Data" levels indicated in Table 2. Then ALE/$\overline{\text{PROG}}$ is pulsed low 25 times as shown in Figure 11.

To program the Encryption Table, repeat the 25-pulse programming sequence for addresses 0 through 1FH, using the "Pgm Encryption Table" levels. Don't forget that after the Encryption Table is programmed, verify cycles will produce only encrypted data.

To program the Lock Bits, repeat the 25-pulse programming sequence using the "Pgm Lock Bit" levels. After one Lock Bit is programmed, further programming of the Code Memory and Encryption Table is disabled. However, the other Lock Bit can still be programmed.

Note that the $\overline{\text{EA}}/V_{PP}$ pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

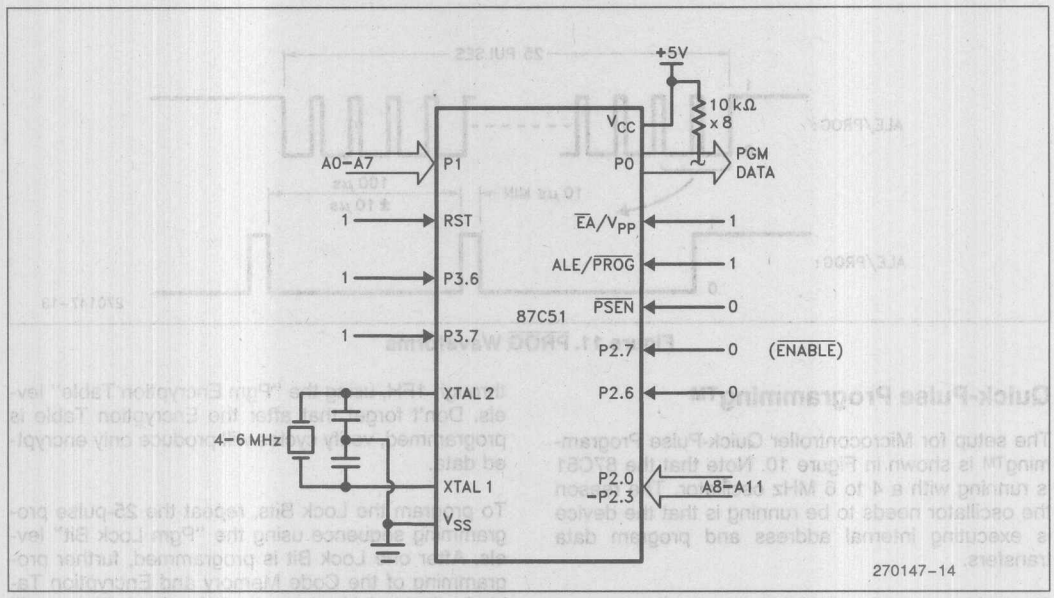


Figure 12. Program Verification

Program Verification

If Lock Bit 2 has not been programmed, the on-chip Program Memory can be read out for program verification. The address of the Program Memory location to be read is applied to Ports 1 and 2 as shown in Figure 12. The other pins are held at the "Verify Code Data" levels indicated in Table 2. The contents of the addressed location will be emitted on Port 0. External pullups are required on Port 0 for this operation. Detailed timing specifications are shown in later sections of this data sheet.

If the Encryption Table has been programmed, the data presented at Port 0 will be the Exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the Encryption Table contents in order to correctly decode the verification data. The Encryption Table itself can not be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values returned are:

(030H) = 89H indicates manufactured by Intel

(031H) = 57H indicates 87C51

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 2, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at least 15 W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000 μW/cm^2 rating for 30 minutes, at a distance of about 1 inch, should be sufficient.

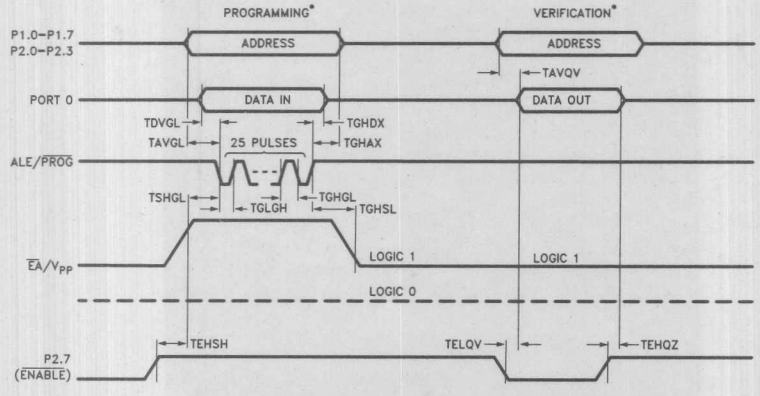
Erasure leaves the array in an all 1s state.

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS:

(T_A = 21°C to 27°C, V_{CC} = 5V ±10%, V_{SS} = 0V)

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	12.5	13.0	V
I_{PP}	Programming Supply Current		50	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{PROG}$ Low	48TCLCL		
TGHAX	Address Hold After $\overline{PROG}$	48TCLCL		
TDVGL	Data Setup to $\overline{PROG}$ Low	48TCLCL		
TGHDX	Data Hold After $\overline{PROG}$	48TCLCL		
TEHSH	P2.7 ($\overline{ENABLE}$) High to V_{PP}	48TCLCL		
TSHGL	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
TGHSL	V_{PP} Hold After $\overline{PROG}$	10		µs
TGLGH	$\overline{PROG}$ Width	90	110	µs
TAVQV	Address to Data Valid		48TCLCL	
TELQV	$\overline{ENABLE}$ Low to Data Valid		48TCLCL	
TEHQZ	Data Float After $\overline{ENABLE}$	0	48TCLCL	
TGHGL	$\overline{PROG}$ High to $\overline{PROG}$ Low	10		µs

EPROM Programming and Verification Waveforms



270147-15

*FOR PROGRAMMING CONDITIONS SEE FIGURE 10.
FOR VERIFICATION CONDITIONS SEE FIGURE 12.

- **Extended Temperature Range**
- **Burn-In**
- **3.5 MHz to 12 MHz V_{CC} = 5V ±10%**

The Intel EXPRESS system offers enhancements to the operational specifications of the MCS®-51 family of microcontrollers. These EXPRESS products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

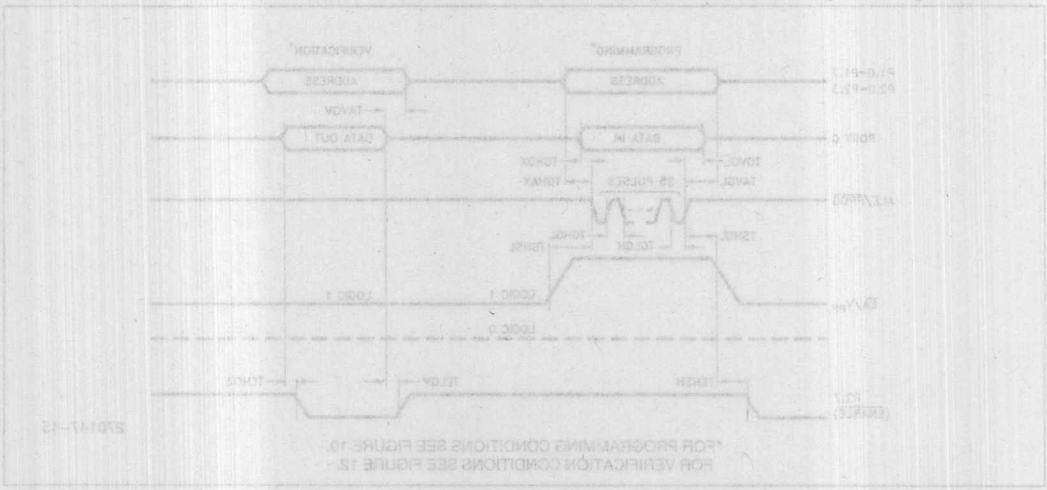
The EXPRESS program includes the commercial standard temperature range with burn-in and an extended temperature range with or without burn-in.

With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to +70°C. With the extended temperature range option, operational characteristics are guaranteed over the range of −40°C to +85°C.

The optional burn-in is dynamic for a minimum time of 160 hours at 125°C with V_{CC} = 6.0V ±0.25V, following guidelines in MIL-STD-883, Method 1015.

Package types and EXPRESS versions are identified by a one- or two-letter prefix to the part number. The prefixes are listed in Table 1.

For the extended temperature range option, this data sheet specifies the parameters which deviate from their commercial temperature range limits. The commercial temperature range data sheets are applicable for all parameters not listed here.



October 1987
Order Number: 270430-001

Electrical Deviations from Commercial Specifications for Extended Temperature Range

D.C. and A.C. parameters not included here are the same as in the commercial temperature range data sheets.

D.C. CHARACTERISTICS $T_A = -40°C$ to $+85°C$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	Limits		Unit	Test Conditions
		Min	Max		
V_{IL}	Input Low Voltage (Except EA)	−0.5	$0.2V_{CC} - 0.15$	V	
V_{IL1}	EA	0	$0.2V_{CC} - 0.35$	V	
V_{IH}	Input High Voltage (Except XTAL1, RST)	$0.2V_{CC} + 1$	$V_{CC} + 0.5$	V	
V_{IH1}	Input High Voltage to XTAL1, RST	$0.7V_{CC} + 0.1$	$V_{CC} + 0.5$	V	
I_{IL}	Logical 0 Input Current (Port 1, 2, 3)		−75	μA	$V_{IN} = 0.45V$
I_{TL}	Logical 1 to 0 transition Current (Ports 1, 2, 3)		−750	μA	$V_{IN} = 2.0V$
I_{CC}	Power Supply Current				(Note 1)
	Active Mode		35	mA	
	Idle Mode		6	mA	
	Power Down Mode		50	μA	

NOTE:
1. $V_{CC} = 4.5V-5.5V$, Frequency Range = 3.5 MHz–12 MHz.

Table 1. Prefix Identification

Prefix	Package Type	Temperature Range(2)	Burn-In(3)
P	Plastic	Commercial	No
D	Cerdip	Commercial	No
N	PLCC	Commercial	No
R	LCC	Commercial	No
TP	Plastic	Extended	No
TD	Cerdip	Extended	No
TN	PLCC	Extended	No
TR	LCC	Extended	No
QP	Plastic	Commercial	Yes
QD	Cerdip	Commercial	Yes
QN	PLCC	Commercial	Yes
QR	LCC	Commercial	Yes
LP	Plastic	Extended	Yes
LD	Cerdip	Extended	Yes
LN	PLCC	Extended	Yes
LR	LCC	Extended	Yes

NOTES:
2. Commercial temperature range is 0°C to +70°C. Extended temperature range is −40°C to +85°C.
3. Burn-in is dynamic for a minimum time of 160 hours at +125°C, V_{CC} = 6.0V ±0.25V, following guidelines in MIL-STD-883 Method 1015 (Test Condition D).

Examples:

P87C51 indicates 87C51 in a plastic package and specified for commercial temperature range, without burn-in.
LD87C51 indicates 87C51 in a cerdip package and specified for extended temperature range with burn-in.

87C51FA (87C252)
CHMOS SINGLE-CHIP 8-BIT MICROCONTROLLER WITH PROGRAMMABLE COUNTER ARRAY, UP/DOWN COUNTER, 8K BYTES USER PROGRAMMABLE EPROM

- High Performance CHMOS EPROM
- Power Control Modes
- Three 16-Bit Timer/Counters
- Programmable Counter Array with:
  - — High Speed Output,
  - — Compare/Capture,
  - — Pulse Width Modulator,
  - — Watchdog Timer capabilities
- Up/Down Timer/Counter
- Two Level Program Lock System
- 8K On-Chip EPROM
- 256 Bytes of On-Chip Data RAM
- Quick Pulse Programming™ Algorithm
- Boolean Processor
- 32 Programmable I/O Lines
- 7 Interrupt Sources
- Programmable Serial Channel with:
  - — Framing Error Detection
  - — Automatic Address Recognition
- TTL Compatible Logic Levels
- 64K External Program Memory Space
- 64K External Data Memory Space
- MCS®-51 Fully Compatible Instruction Set
- Power Saving Idle and Power Down Modes
- ONCE™ (On-Circuit Emulation) Mode

MEMORY ORGANIZATION

PROGRAM MEMORY: Up to 8K bytes of the program memory can reside in the on-chip EPROM. In addition the device can address up to 64K of program memory external to the chip.

DATA MEMORY: This microcontroller has a 256 x 8 on-chip RAM. In addition it can address up to 64K bytes of external data memory.

The Intel 87C51FA is a single-chip control oriented microcontroller which is fabricated on Intel's reliable CHMOS II-E technology. Being a member of the MCS®-51 family, the 87C51FA uses the same powerful instruction set, has the same architecture, and is pin for pin compatible with the existing MCS-51 products. The 87C51FA is an enhanced version of the 87C51. It's added features make it an even more powerful microcontroller for applications that require Pulse Width Modulation, High Speed I/O, and up/down counting capabilities such as motor control. It also has a more versatile serial channel that facilitates multi-processor communications.

October 1987
Order Number: 270258-002

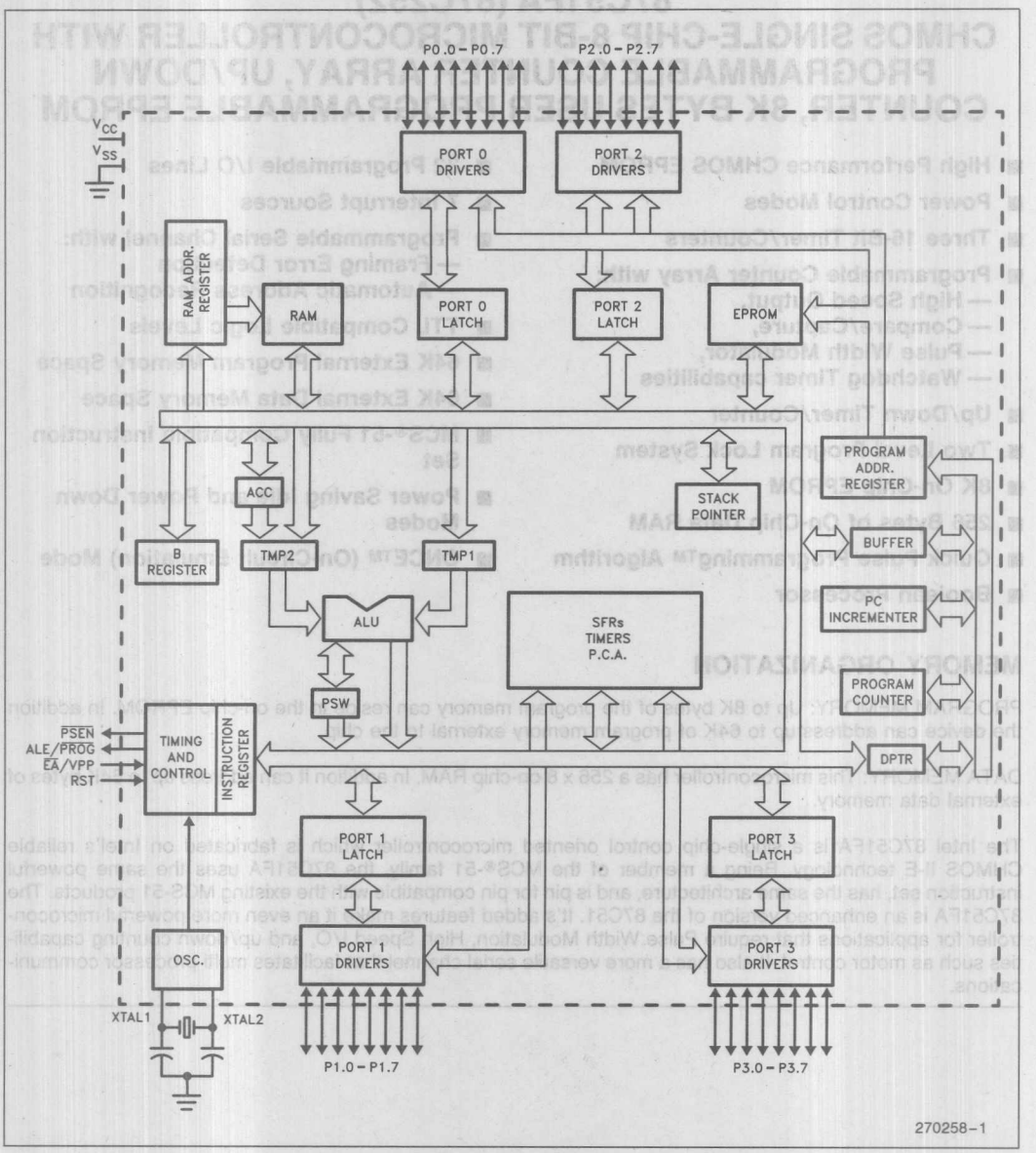


Figure 1. 87C51FA Block Diagram

PIN DESCRIPTIONS

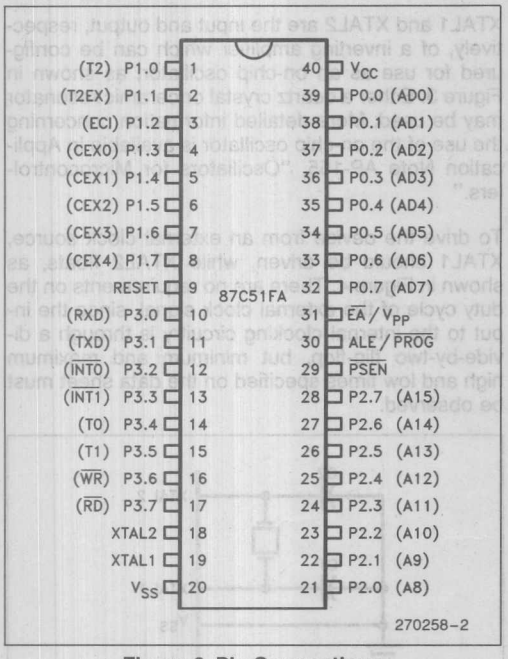


Figure 2. Pin Connections

V_{CC}: Supply voltage.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit, open drain, bidirectional I/O port. As an output port each pin can sink several LS TTL inputs. Port 0 pins that have 1's written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting1's, and can source and sink several LS TTL inputs.

Port 0 also receives the code bytes during EPROM programming, and outputs the code bytes during program verification. External pullup resistors are required during program verification.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can drive LS TTL inputs. Port 1 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

In addition, Port 1 serves the functions of the following special features of the 87C51FA:

Port Pin	Alternate Function
P1.0	T2 (External Count Input to Timer/ Counter 2)
P1.1	T2EX (Timer/Counter 2 Capture/ Reload Trigger and Direction Control)
P1.2	ECI (External Count Input to the PCA)
P1.3	CEX0 (External I/O for Compare/ Capture Module 0)
P1.4	CEX1 (External I/O for Compare/ Capture Module 1)
P1.5	CEX2 (External I/O for Compare/ Capture Module 2)
P1.6	CEX3 (External I/O for Compare/ Capture Module 3)
P1.7	CEX4 (External I/O for Compare/ Capture Module 4)

Port 1 receives the low-order address bytes during EPROM programming and verifying.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can drive LS TTL inputs. Port 2 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1's. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Some Port 2 pins receive the high-order address bits during EPROM programming and program verification.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can drive LS TTL inputs. Port 3 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. An internal pulldown resistor permits a power-on reset with only a capacitor connected to V_{CC}.

ALE: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin (ALE/$\overline{PROG}$) is also the program pulse input during EPROM programming for the 87C51FA.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

Throughout the remainder of this data sheet, ALE will refer to the signal coming out of the ALE/$\overline{PROG}$ pin, and the pin will be referred to as the ALE/$\overline{PROG}$ pin.

$\overline{PSEN}$: Program Store Enable is the read strobe to external Program Memory.

When the 87C51FA is executing code from external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external Data Memory.

$\overline{EA}$/V_{PP}: External Access enable. $\overline{EA}$ must be strapped to VSS in order to enable the device to fetch code from external Program Memory locations 0000H to 1FFFH. Note, however, that if either of the Program Lock bits are programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the programming supply voltage (V_{PP}) during EPROM programming.

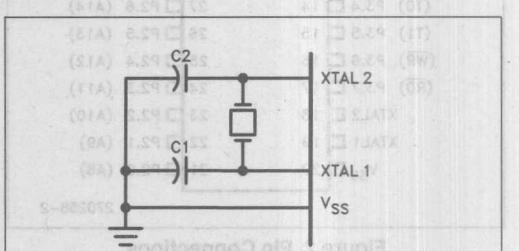
XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of a inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 floats, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the data sheet must be observed.



C1, C2 = 30 pF ±10 pF for Crystals
= 10 pF for Ceramic Resonators

Figure 3. Oscillator Connections

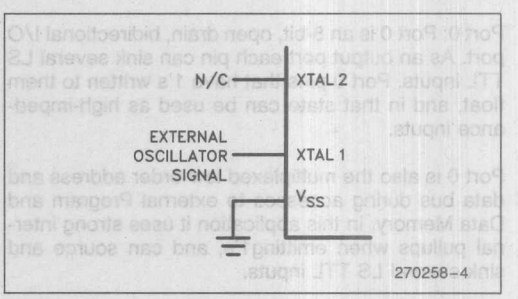


Figure 4. External Clock Drive Configuration

IDLE MODE

The user's software can invoke the Idle Mode. When the microcontroller is in this mode, power consumption is reduced. The Special Function Registers and the onboard RAM retain their values during Idle, but the processor stops executing instructions. Idle Mode will be exited if the chip is reset or if an enabled interrupt occurs. The PCA timer/counter can optionally be left running or paused during Idle Mode.

POWER DOWN MODE

To save even more power, a Power Down mode can be invoked by software. In this mode, the oscillator is stopped and the instruction that invoked Power Down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the Power Down mode is terminated.

On the 87C51FA either a hardware reset or an external interrupt can cause an exit from Power Down. Reset redefines all the SFRs but does not change the on-chip RAM. An external interrupt allows both the SFRs and on-chip RAM to retain their values. The interrupt must be enabled and configured as level sensitive. To properly terminate Power Down the reset or external interrupt should not be executed before V_{CC} is restored to its normal operating level, and must be held active long enough for the oscillator to restart and stabilize.

DESIGN CONSIDERATION

- Ambient light is known to affect the internal RAM contents during operation. If the 87C51FA application requires the part to be run under ambient lighting, an opaque label should be placed over the window to exclude light.
- When the idle mode is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

ONCE™ MODE

The ONCE ("On-Circuit Emulation") Mode facilitates testing and debugging of systems using the 87C51FA without the 87C51FA having to be removed from the circuit. The ONCE Mode is invoked by:

1) Pull ALE low while the device is in reset and $\overline{PSEN}$ is high;
2) Hold ALE low as RST is deactivated.

While the device is in ONCE Mode, the Port 0 pins go into a float state, and the other port pins and ALE and $\overline{PSEN}$ are weakly pulled high. The oscillator circuit remains active. While the 87C51FA is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored when a normal reset is applied.

Table 1. Status of the External Pins during Idle and Power Down

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data

NOTE:
For more detailed information on the reduced power modes refer to current Embedded Controller Handbook, and Application Note AP-252, "Designing with the 80C51BH."

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on EA/V_{PP} Pin to V_{SS} . . . . . 0V to +13.0V
Voltage on Any Other Pin to V_{SS} . . −0.5V to +6.5V
Power Dissipation 1.5W
(based on PACKAGE heat transfer limitations, not device power consumption)

Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

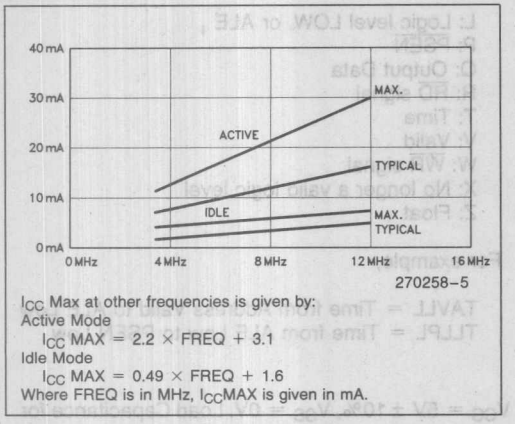
NOTICE: Specifications contained within the following tables are subject to change.

ADVANCED INFORMATION—CONTACT INTEL FOR DESIGN-IN INFORMATION

D.C. CHARACTERISTICS: (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V)

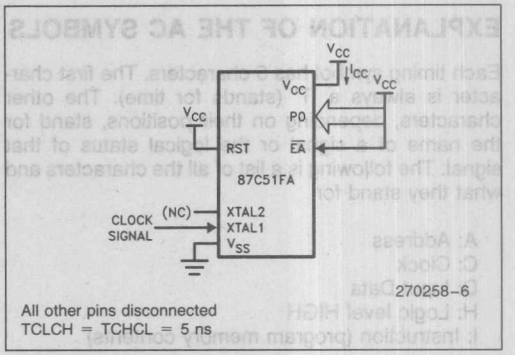
Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.2 V_{CC} − 0.1	V	
V_{IL1}	Input Low Voltage $\overline{EA}$	0	0.2 V_{CC} − 0.3	V	
V_{IH}	Input High Voltage (Except XTAL2, RST, $\overline{EA}$)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage (XTAL, RST)	0.7 V_{CC}	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage (Ports 1, 2 and 3)		0.45	V	I_{OL} = 1.6 mA(1)
V_{OL1}	Output Low Voltage (Port 0, ALE/$\overline{PROG}$, $\overline{PSEN}$)		0.45	V	I_{OL} = 3.2 mA(1)
V_{OH}	Output High Voltage (Ports 1, 2 and 3 ALE/$\overline{PROG}$ and $\overline{PSEN}$)	2.4		V	I_{OH} = −60 μA
		0.9 V_{CC}		V	I_{OH} = −10 μA(2)
V_{OH1}	Output High Voltage	2.4		V	I_{OH} = −800 μA
	(Port 0 in External Bus Mode)	0.9 V_{CC}		V	I_{OH} = −80 μA(2)
I_{IL}	Logical 0 Input Current (Ports 1, 2, and 3)		−50	μA	V_{IN} = 0.45V
I_{LI}	Input leakage Current (Port 0 and $\overline{EA}$)		±10	μA	V_{IN} = V_{IL} or V_{IH}
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, and 3)		−650	μA	V_{IN} = 2V
RRST	RST Pulldown Resistor	40	225	KΩ	
CIO	Pin Capacitance		10	pF	@1MHz, 25°C
I_{CC}	Power Supply Current: Running at 12 MHz (Figure 5) Idle Mode at 12 MHz (Figure 5) Power Down Mode		 30 7.5 100	 mA mA μA	(Note 3)

NOTES:
1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1 to 0 transitions during bus operations. In applications where capacitance loading exceeds 100 pFs, the noise pulse on the ALE signal may exceed 0.8V. In these cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an Address Latch with a Schmitt Trigger Strobe input.
2. Capacitive loading on Ports 0 and 2 cause the V_{OH} on ALE and $\overline{PSEN}$ to drop below the 0.9 V_{CC} specification when the address lines are stabilizing.
3. See Figures 6–9 for test conditions.



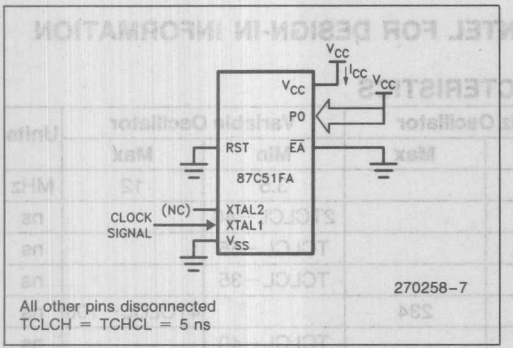
I_{CC} Max at other frequencies is given by:
Active Mode
I_{CC} MAX = 2.2 × FREQ + 3.1
Idle Mode
I_{CC} MAX = 0.49 × FREQ + 1.6
Where FREQ is in MHz, I_{CC}MAX is given in mA.

Figure 5. I_{CC} vs Frequency



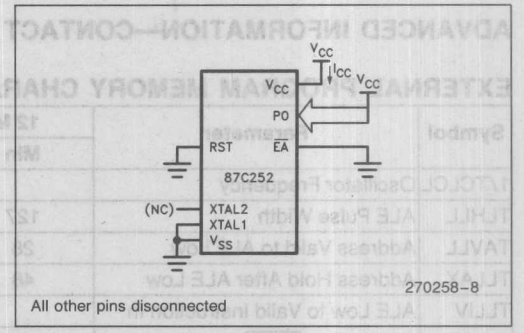
All other pins disconnected
TCLCH = TCHCL = 5 ns

Figure 6. I_{CC} Test Condition, Active Mode



All other pins disconnected
TCLCH = TCHCL = 5 ns

Figure 7. I_{CC} Test Condition Idle Mode



All other pins disconnected

Figure 9. I_{CC} Test Condition, Power Down Mode. V_{CC} = 2.0V to 5.5V.

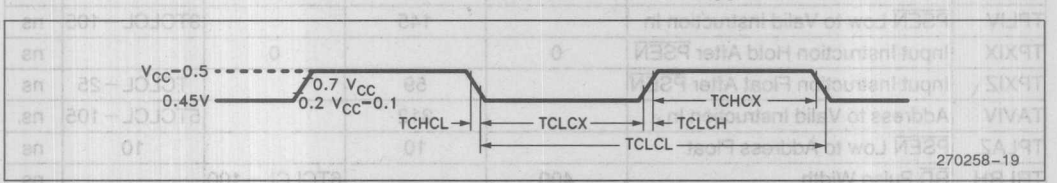


Figure 8. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes. TCLCH = TCHCL = 5 ns.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

A: Address
C: Clock
D: Input Data
H: Logic level HIGH
I: Instruction (program memory contents)
L: Logic level LOW, or ALE
P: $\overline{\text{PSEN}}$
Q: Output Data
R: $\overline{\text{RD}}$ signal
T: Time
V: Valid
W: $\overline{\text{WR}}$ signal
X: No longer a valid logic level
Z: Float

For example,

TAVLL = Time from Address Valid to ALE Low
TLLPL = Time from ALE Low to PSEN Low

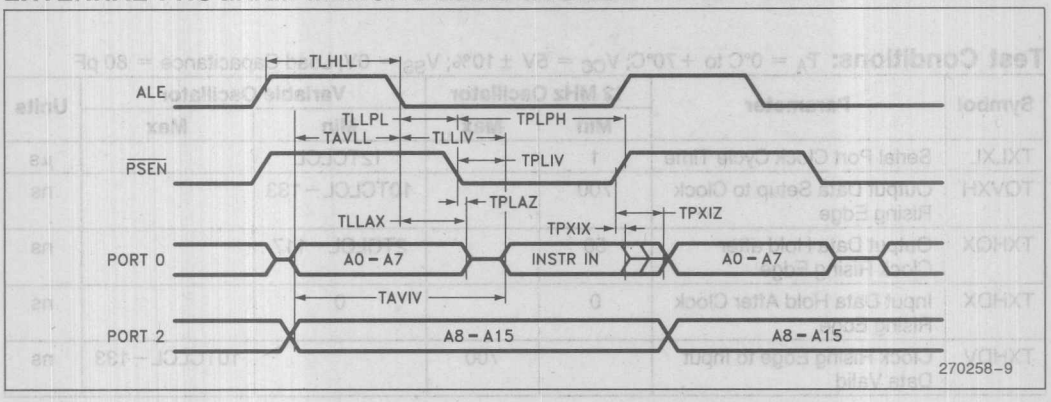
A.C. CHARACTERISTICS (T_A = 0°C to +70°C, V_{CC} = 5V ±10%, V_{SS} = 0V, Load Capacitance for Port 0, ALE/$\overline{\text{PROG}}$ and PSEN = 100 pF, Load Capacitance for All Other Outputs = 80 pF)

ADVANCED INFORMATION—CONTACT INTEL FOR DESIGN-IN INFORMATION

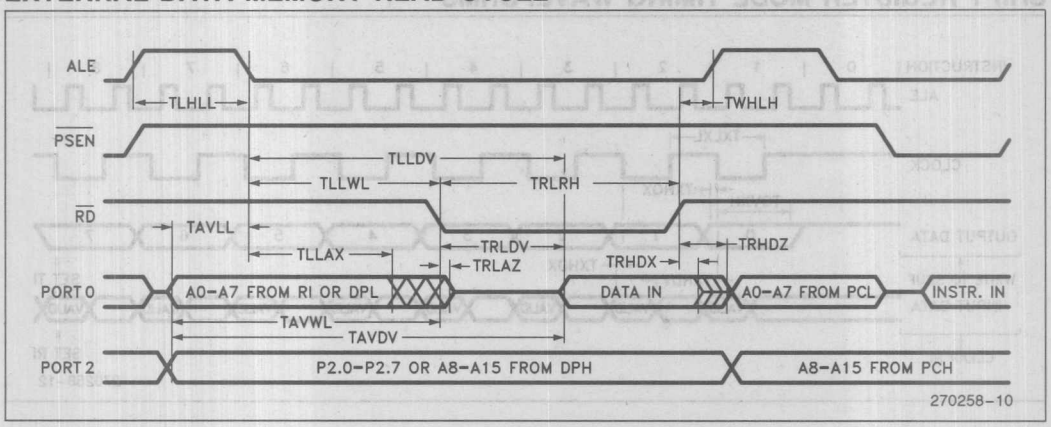
EXTERNAL PROGRAM MEMORY CHARACTERISTICS

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12	MHz
TLHLL	ALE Pulse Width	127		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	28		TCLCL−55		ns
TLLAX	Address Hold After ALE Low	48		TCLCL−35		ns
TLLIV	ALE Low to Valid Instruction In		234		4TCLCL−100	ns
TLLPL	ALE Low to $\overline{\text{PSEN}}$ Low	43		TCLCL−40		ns
TPLPH	$\overline{\text{PSEN}}$ Pulse Width	205		3TCLCL−45		ns
TPLIV	$\overline{\text{PSEN}}$ Low to Valid Instruction In		145		3TCLCL−105	ns
TPXIX	Input Instruction Hold After $\overline{\text{PSEN}}$	0		0		ns
TPXIZ	Input Instruction Float After $\overline{\text{PSEN}}$		59		TCLCL−25	ns
TAVIV	Address to Valid Instruction In		312		5TCLCL−105	ns
TPLAZ	$\overline{\text{PSEN}}$ Low to Address Float		10		10	ns
TRLRH	$\overline{\text{RD}}$ Pulse Width	400		6TCLCL−100		ns
TWLWH	$\overline{\text{WR}}$ Pulse Width	400		6TCLCL−100		ns
TRLDV	$\overline{\text{RD}}$ Low to Valid Data In		252		5TCLCL−165	ns
TRHDX	Data Hold After $\overline{\text{RD}}$	0		0		ns
TRHDZ	Data Float After $\overline{\text{RD}}$		97		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL−150	ns
TAVDV	Address to Valid Data In		585		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{\text{RD}}$ or $\overline{\text{WR}}$ Low	200	300	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address Valid to $\overline{\text{WR}}$ Low	203		4TCLCL−130		ns
TQVWX	Address Valid before $\overline{\text{WR}}$	23		TCLCL−60		ns
TWHQX	Data Hold after $\overline{\text{WR}}$	33		TCLCL−50		ns
TRLAZ	$\overline{\text{RD}}$ Low to Address Float		0		0	ns
TWHLH	$\overline{\text{RD}}$ or $\overline{\text{WR}}$ High to ALE High	43	123	TCLCL−40	TCLCL+40	ns

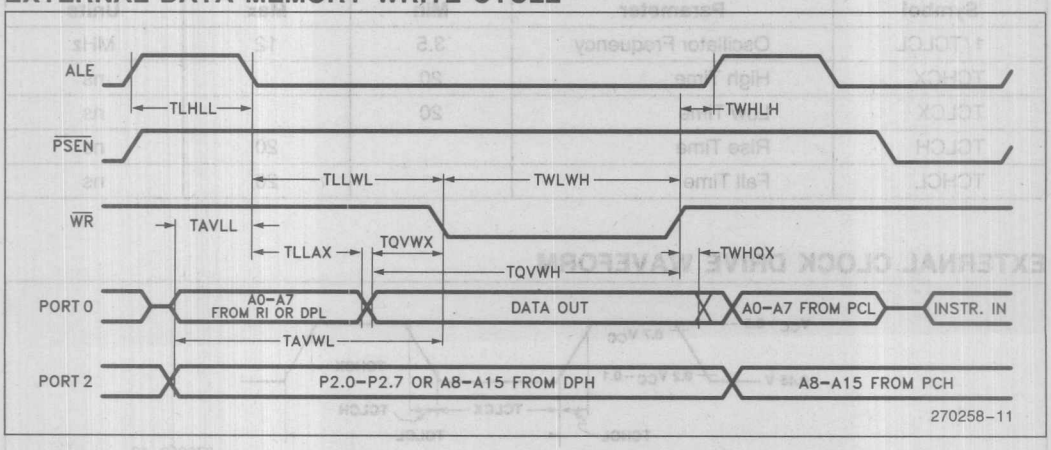
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE

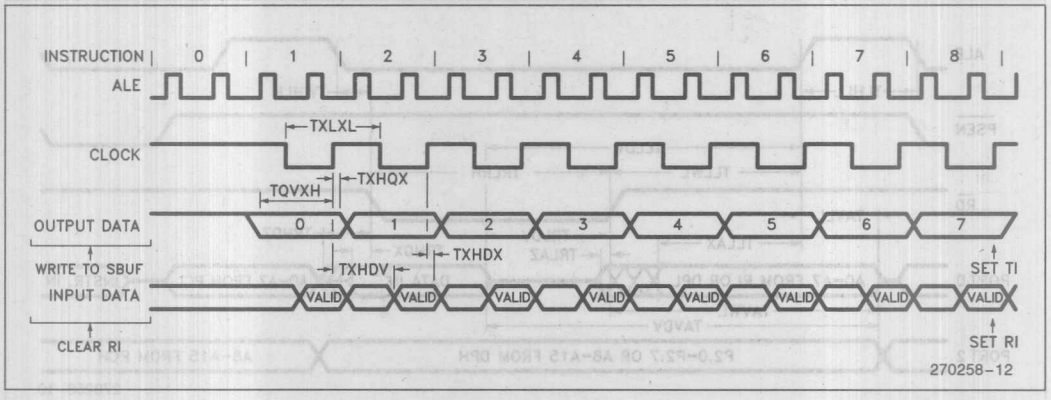


SERIAL PORT TIMING - SHIFT REGISTER MODE

Test Conditions: T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1		12TCLCL		µs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL−133		ns
TXHQX	Output Data Hold after Clock Rising Edge	50		2TCLCL−117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL−133	ns

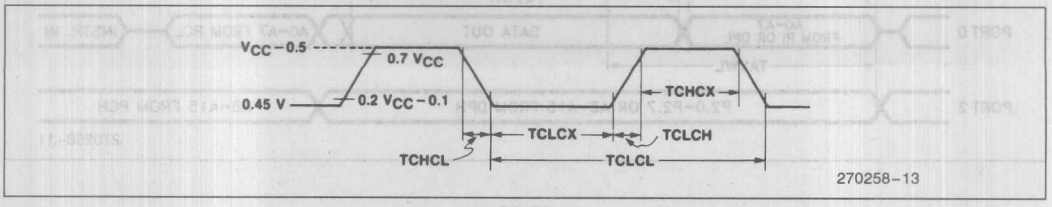
SHIFT REGISTER MODE TIMING WAVEFORMS



EXTERNAL CLOCK DRIVE

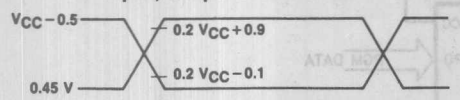
Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

EXTERNAL CLOCK DRIVE WAVEFORM



A.C. TESTING INPUT

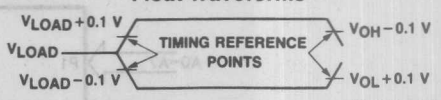
Input, Output Waveforms



270258-14

AC Inputs during testing are driven at $V_{CC}-0.5V$ for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at V_{IH} min for a Logic "1" and V_{OL} max for a Logic "0".

Float Waveforms



270258-15

For timing purposes a port pin is no longer floating when a 100 mV change from load voltage occurs, and begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OL}/I_{OH} \geq \pm 20$ mA.

EPROM CHARACTERISTICS

Table 2 shows the logic levels for programming the Program Memory, the Encryption Table, and the Lock Bits and for reading the signature bytes.

Table 2. EPROM Programming Modes

Mode	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.6	P3.7
Program Code Data	1	0	0*	V_{PP}	1	0	1	1
Verify Code Data	1	0	1	1	0	0	1	1
Program Encryption Table Use Addresses 0–1FH	1	0	0*	V_{PP}	1	0	0	1
Program Lock x=1	1	0	0*	V_{PP}	1	1	1	1
Bits (LBx) x=2	1	0	0*	V_{PP}	1	1	0	0
Read Signature	1	0	1	1	0	0	0	0

NOTES:
"1" = Valid high for that pin
"0" = Valid low for that pin
"VPP" = +12.75V ±0.25V
* ALE/$\overline{PROG}$ is pulsed low for 100 μs for programming. (Quick-Pulse Programming™)

PROGRAMMING THE EPROM

To be programmed, the part must be running with a 4 to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate internal EPROM locations.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0 - P2.4 of Port 2, while the code byte to be programmed into that location is applied to Port 0. The other Port 2 and 3 pins, RST $\overline{PSEN}$, and $\overline{EA}/V_{PP}$ should be held at the "Program" levels indicated in Table 2. ALE/$\overline{PROG}$ is pulsed low to program the code byte into the addressed EPROM location. The setup is shown in Figure 10.

Normally $\overline{EA}/V_{PP}$ is held at logic high until just before ALE/$\overline{PROG}$ is to be pulsed. Then $\overline{EA}/V_{PP}$ is raised to V_{PP}, ALE/$\overline{PROG}$ is pulsed low, and then $\overline{EA}/V_{PP}$ is returned to a valid high voltage. The voltage on the $\overline{EA}/V_{PP}$ pin must be at the valid $\overline{EA}/V_{PP}$ high level before a verify is attempted. Waveforms and detailed timing specifications are shown in later sections of this data sheet.

Note that the $\overline{EA}/V_{PP}$ pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches.

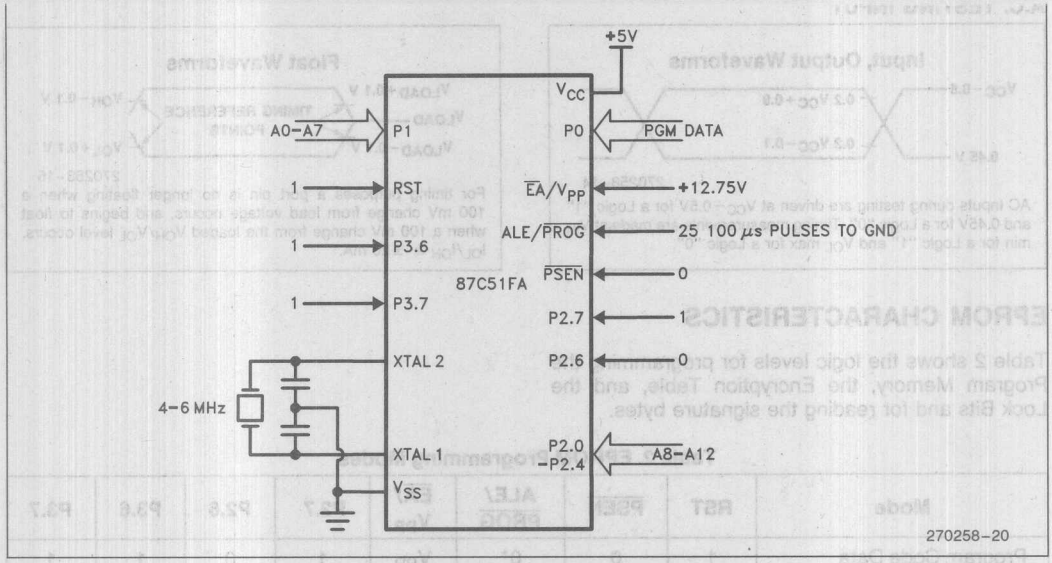


Figure 10. Programming the EPROM

Quick-Pulse Programming™ Algorithm

The 87C51FA can be programmed using the Quick-Pulse Programming™ Algorithm for microcontrollers. The features of the new programming method are a lower V_{PP} (12.75V as compared to 21V) and a shorter programming pulse. It is possible to program the entire 8K Bytes of EPROM memory in less than 25 seconds with this algorithm!

To program the part using the new algorithm, V_{PP} must be 12.75V ±0.25V. ALE/$\overline{PROG}$ is pulsed low for 100 μs, 25 times as shown in Figure 11. Then, the byte just programmed may be verified. After programming, the entire array should be verified. The Program Lock features are programmed using the same method, but with the setup as shown in Table 2. The only difference in programming Program Lock features is that the Program Lock features cannot be directly verified. Instead, verification of programming is by observing that their features are enabled.

Program Verification

If the Program Lock Bits have not been programmed, the on-chip Program Memory can be read out for verification purposes, if desired, either during or after the programming operation. The address of the Program Memory location to be read is applied to Port 1 and pins P2.0 - P2.4. The other pins should be held at the "Verify" levels indicated in Table 3. The contents of the addressed locations will come out on Port 0. External pullups are required on Port 0 for this operation.

If the Encryption Array in the EPROM has been programmed, the data present at Port 0 will be Code Data XNOR Encryption Data. The user must know the Encryption Array contents to manually "unencrypt" the data during verify.

The setup, which is shown in Figure 12, is the same as for programming the EPROM except that pin P2.7 is held at a logic low, or may be used as an active-low read strobe.

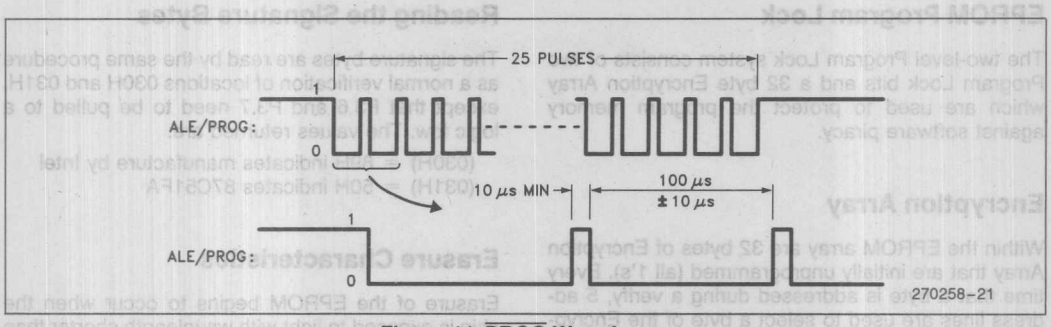


Figure 11. $\overline{PROG}$ Waveforms

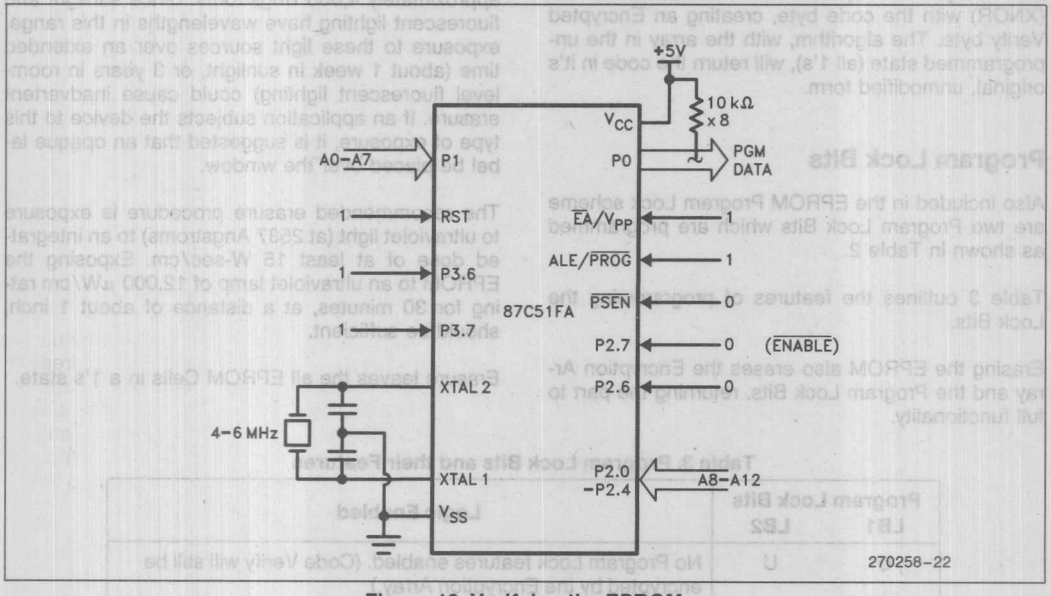


Figure 12. Verifying the EPROM

EPROM Program Lock

The two-level Program Lock system consists of two Program Lock bits and a 32 byte Encryption Array which are used to protect the program memory against software piracy.

Encryption Array

Within the EPROM array are 32 bytes of Encryption Array that are initially unprogrammed (all 1's). Every time that a byte is addressed during a verify, 5 address lines are used to select a byte of the Encryption Array. This byte is then exclusive-NOR'ed (XNOR) with the code byte, creating an Encrypted Verify byte. The algorithm, with the array in the unprogrammed state (all 1's), will return the code in it's original, unmodified form.

Program Lock Bits

Also included in the EPROM Program Lock scheme are two Program Lock Bits which are programmed as shown in Table 2.

Table 3 outlines the features of programming the Lock Bits.

Erasing the EPROM also erases the Encryption Array and the Program Lock Bits, returning the part to full functionality.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values returned are:

(030H) = 89H indicates manufacture by Intel
(031H) = 50H indicates 87C51FA

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelength shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at least 15 W-sec/cm. Exposing the EPROM to an ultraviolet lamp of 12,000 μW/cm rating for 30 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the all EPROM Cells in a 1's state.

Table 3. Program Lock Bits and their Features

Program Lock Bits LB1	LB2	Logic Enabled
U	U	No Program Lock features enabled. (Code Verify will still be encrypted by the Encryption Array.)
P	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the EPROM is disabled.
P	P	Same as above, but Verify is also disabled
U	P	Reserved for Future Definition

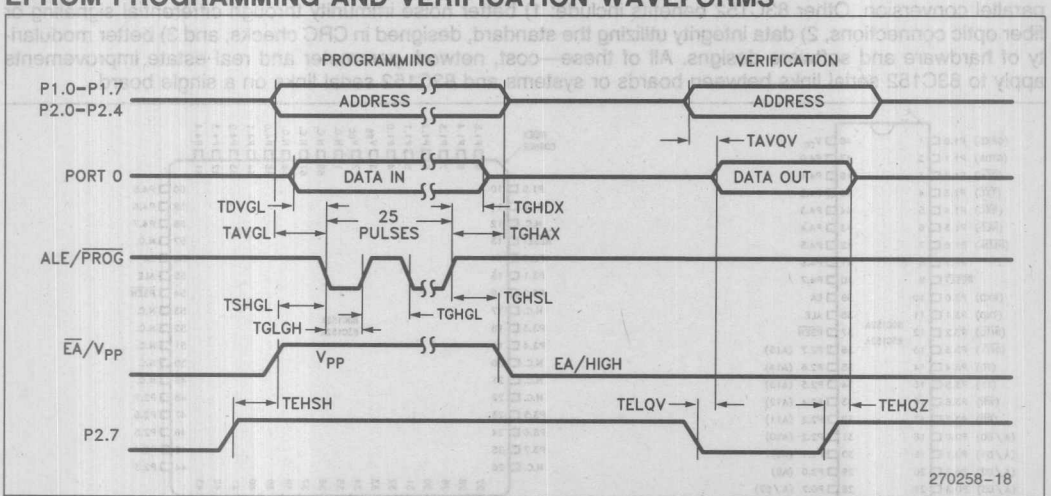
EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

(T_A = 21°C to 27°C; V_{CC} = 5V ± 0.25V; V_{SS} = 0V)

ADVANCED INFORMATION—CONTACT INTEL FOR DESIGN-IN INFORMATION

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	12.5	13.0	V
I_{PP}	Programming Supply Current		50	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{PROG}$ Low	48TCLCL		
TGHAX	Address Hold after $\overline{PROG}$	48TCLCL		
TDVGL	Data Setup to $\overline{PROG}$ Low	48TCLCL		
TGHDX	Data Hold after $\overline{PROG}$	48TCLCL		
TEHSH	P2.7 (ENABLE) High to V_{PP}	48TCLCL		
TSHGL	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
TGHSL	V_{PP} Hold after $\overline{PROG}$	10		µs
TGLGH	$\overline{PROG}$ Width	90	110	µs
TAVQV	Address to Data Valid		48TCLCL	
TELQV	ENABLE Low to Data Valid		48TCLCL	
TEHQZ	Data Float after ENABLE	0	48TCLCL	
TGHGL	$\overline{PROG}$ High to $\overline{PROG}$ Low	10		µs

EPROM PROGRAMMING AND VERIFICATION WAVEFORMS



83C152A
UNIVERSAL COMMUNICATION CONTROLLER 8-BIT MICROCOMPUTER WITH FACTORY MASK PROGRAMMABLE ROM

80C152A
UNIVERSAL COMMUNICATION CONTROLLER 8-BIT MICROCOMPUTER

- **Superset of 80C51BH Architecture**
- **Multi-Protocol Serial Communication I/O Port (1.5 Mbps/2.4 Mbps Max)**
  - **— SDLC**
  - **— HDLC**
  - **— CSMA/CD**
  - **— User Definable Protocols**
- **Full Duplex/Half Duplex**
- **MCS®-51 Compatible UART**
- **12 MHz Maximum Clock Frequency**
- **Multiple Power Conservation Modes**
- **64KB Program Memory Addressing**
- **64KB Data Memory Addressing**
- **256 Bytes On-Chip RAM**
- **Dual On-Chip DMA Channels**
- **Hold/Hold Acknowledge**
- **Two General Purpose Timer/Counters**
- **56 Special Function Registers**
- **11 Interrupt Sources**
- **Available in 48 Pin Dual-in-Line Package and 68 Pin Surface Mount PLCC Package**
  (See Packaging Spec. Order #231369)

The 80C152, which is based on the MCS®-51 CPU, is a highly integrated single-chip 8-bit microcontroller designed for cost-sensitive, high-speed, serial communications. It is well suited for implementing Integrated Services Digital Networks (ISDN), emerging Local Area Networks, and user defined serial backplane applications. In addition to the multi-protocol communication capability, the 80C152 offers traditional microcontroller features for peripheral I/O interface and control.

Silicon implementations are much more cost effective than multiwire cables found in board level parallel-to-serial and serial-to-parallel converters. The 83C152 contains, in silicon, all the features needed for the serial-to-parallel conversion. Other 83C152 benefits include: 1) better noise immunity through differential signaling or fiber optic connections, 2) data integrity utilizing the standard, designed in CRC checks, and 3) better modularity of hardware and software designs. All of these—cost, network parameter and real estate improvements apply to 83C152 serial links between boards or systems and 83C152 serial links on a single board.

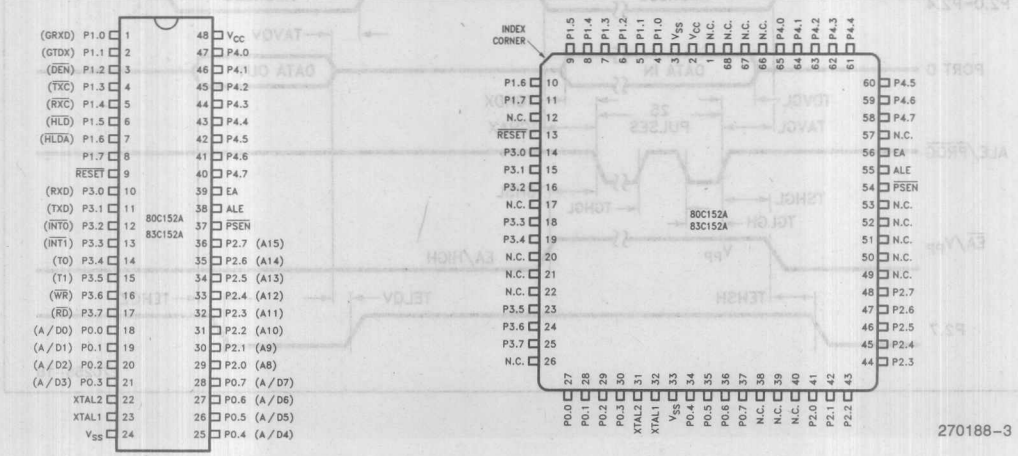


Figure 1. Connection Diagrams

September 1987
Order Number: 270188-003

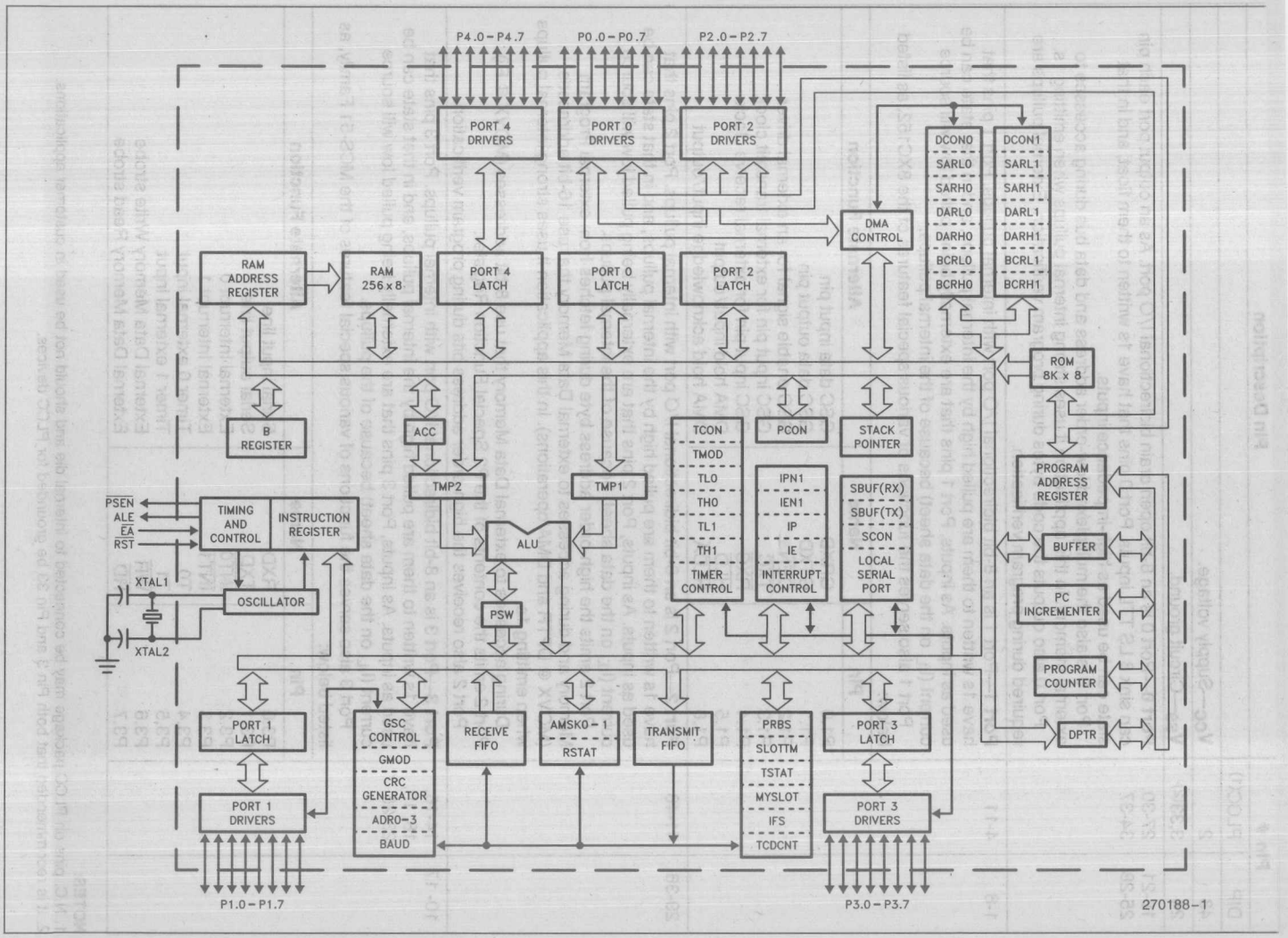


Figure 2. Block Diagram

Pin #		Pin Description
DIP	PLCC(1)	
48	2	V_{CC}—Supply voltage.
24	3,33(2)	V_{SS}—Circuit ground.
18-21, 25-28	27-30, 34-37	**Port 0**—Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external memory. In this application it uses strong internal pullups when emitting 1s. Port 0 also outputs the code bytes during program verification. External pullups are required during program verification.
1-8	4-11	**Port 1**—Port 1 is an 8-bit bidirectional I/O port with internal pullups. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups. Port 1 also serves the functions of various special features of the 8XC152, as listed below:

Pin	Name	Alternate Function
P1.0	GRXD	GSC data input pin
P1.1	GTXD	GSC data output pin
P1.2	$\overline{DEN}$	GSC enable signal for an external driver
P1.3	$\overline{TXC}$	GSC input pin for external transmit clock
P1.4	$\overline{RXC}$	GSC input pin for external receive clock
P1.5	$\overline{HLD}$	DMA hold input/output
P1.6	$\overline{HLDA}$	DMA hold acknowledge input/output

29-36	41-48	**Port 2**—Port 2 is an 8-bit bidirectional I/O port with internal pullups. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups. Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @ DPTR and DMA operations). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @ Ri), Port 2 emits the contents of the P2 Special Function Register. Port 2 also receives the high-order address bits during program verification.
10- 17	14-16, 18, 19, 23-25	**Port 3**—Port 3 is an 8-bit bidirectional I/O port with internal pullups. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the pullups. Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Pin	Name	Alternate Function
P3.0	RXD	Serial input line
P3.1	TXD	Serial output line
P3.2	$\overline{INT0}$	External Interrupt 0
P3.3	$\overline{INT1}$	External Interrupt 1
P3.4	T0	Timer 0 external input
P3.5	T1	Timer 1 external input
P3.6	$\overline{WR}$	External Data Memory Write strobe
P3.7	$\overline{RD}$	External Data Memory Read strobe

NOTES:
1. N.C. pins on PLCC package may be connected to internal die and should not be used in customer applications.
2. It is recommended that both Pin 3 and Pin 33 be grounded for PLCC devices.

Pin Description (Continued)

Pin #		Pin Description
47-40	65-58	**Port 4**—Port 4 is an 8-bit bidirectional I/O port with internal pullups. Port 4 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 4 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups. In addition, Port 4 also receives the low-order address bytes during program verification.
9	13	**$\overline{RST}$**—Reset input. A logic low on this pin for three machine cycles while the oscillator is running resets the device. An internal pullup resistor permits a power-on reset to be generated using only an external capacitor to V_{SS}. Although the GSC recognizes the reset after three machine cycles, data may continue to be transmitted for up to 4 machine cycles after Reset is first applied.
38	55	**ALE**—Address Latch Enable output signal for latching the low byte of the address during accesses to external memory. In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory. While in Reset, ALE remains at a constant high level.
37	54	**$\overline{PSEN}$**—Program Store Enable is the Read strobe to External Program Memory. When the 8XC152 is executing from external program memory, $\overline{PSEN}$ is active (low). When the device is executing code from External Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to External Data Memory. While in Reset, $\overline{PSEN}$ remains at a constant high level.
39	56	**$\overline{EA}$**—External Access enable. $\overline{EA}$ must be externally pulled low in order to enable the 8XC152 to fetch code from External Program Memory locations 0000H to 0FFFH. $\overline{EA}$ must be connected to V_{CC} for internal program execution.
23	32	**XTAL1**—Input to the inverting oscillator amplifier and input to the internal clock generating circuits.
22	31	**XTAL2**—Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3.

To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left unconnected, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

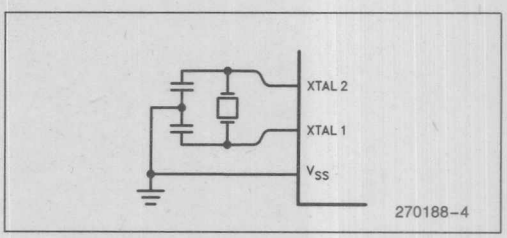


Figure 3. Using the On-Chip Oscillator

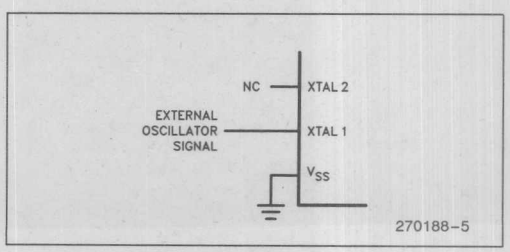


Figure 4. External Clock Drive

IDLE MODE

In Idle Mode, the CPU puts itself to sleep while most of the on-chip peripherals remain active. The major peripherals that do not remain active during Idle, are the DMA channels. The Idle Mode is invoked by software. The content of the on-chip RAM and all the Special Function Registers remain unchanged during this mode. The Idle Mode can be terminated by any enabled interrupt or by a hardware reset.

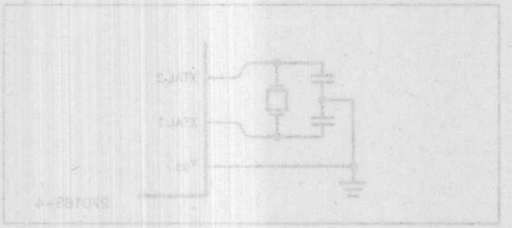
POWER DOWN MODE

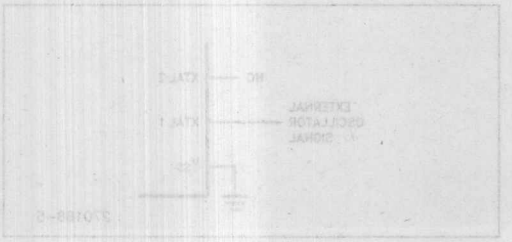
In Power Down Mode, the oscillator is stopped and all on-chip functions cease except that the on-chip RAM contents are maintained. The mode Power Down is invoked by software. The Power Down Mode can be terminated only by a hardware reset.

Table 1. Status of the external pins during Idle and Power Down modes

Mode	Program Memory	ALE	$\overline{PSEN}$	Port 0	Port 1	Port 2	Port 3	Port 4
Idle	Internal	1	1	Data	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data	Data
Power Down	Internal	0	0	Data	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data	Data

NOTE:
For more detailed information on the reduced power modes refer to the Embedded Controller Handbook, and Application Note AP-252, "Designing with the 80C51BH."





ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any pin to V_{SS} . . −0.5V to (V_{CC} + 0.5V)
Voltage on V_{CC} to VSS . . . . . −0.5V to +6.5V
Power Dissipation . . . . . 1.0 W(7)

Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

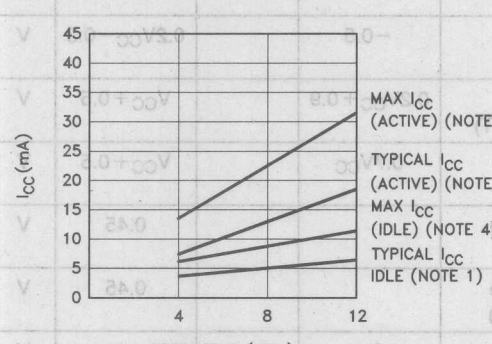
NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V)

Symbol	Parameter	Min	Typ (Note 1)	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage (All Except $\overline{EA}$)	−0.5		$0.2V_{CC}-0.1$	V	
V_{IL1}	Input Low Voltage ($\overline{EA}$)	−0.5		$0.2V_{CC}-0.3$	V	
V_{IH}	Input High Voltage (Except XTAL1, $\overline{RST}$)	$0.2V_{CC}+0.9$		$V_{CC}+0.5$	V	
V_{IH1}	Input High Voltage (XTAL1, $\overline{RST}$)	$0.7V_{CC}$		$V_{CC}+0.5$	V	
V_{OL}	Output Low Voltage (Ports 1, 2, 3, 4)			0.45	V	I_{OL} = 1.6 mA (Note 2)
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$)			0.45	V	I_{OL} = 3.2 mA (Note 2)
V_{OH}	Output High Voltage (Ports 1, 2, 3, 4, ALE, $\overline{PSEN}$)	2.4			V	I_{OH} = −60 μA V_{CC} = 5V ±10%
		$0.9V_{CC}$			V	I_{OH} = −10 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4			V	I_{OH} = −400 μA V_{CC} = 5V ±10%
		$0.9V_{CC}$			V	I_{OH} = −40 μA (Note 3)
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3, 4)			−50	μA	V_{IN} = 0.45V
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, 3, 4)			−650	μA	V_{IN} = 2V
I_{LI}	Input Leakage (Port 0, $\overline{EA}$)			±10	μA	$0.45 < V_{IN} < V_{CC}$
RRST	Reset Pullup Resistor	40			kΩ	

NOTES:
1. "Typicals" are based on samples taken from early manufacturing lots and are not guaranteed. The measurements were made with V_{CC} = 5V at room temperature.
2. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
3. Capacitive loading on Ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. I_{CC} is measured with all output pins disconnected; XTAL1 driven with TCLCH, TCHCL = 5 ns, $V_{IL} = V_{SS} + 0.5V$, $V_{IH} = V_{CC} - 0.5V$; XTAL2 N.C.; Port 0 pins connected to V_{CC}. "Operating" current is measured with $\overline{EA}$ connected to V_{CC} and $\overline{RST}$ connected to V_{SS}. "Idle" current is measured with $\overline{EA}$ connected to V_{SS}, $\overline{RST}$ connected to V_{CC} and GSC inactive.
5. The specifications relating to external data memory characteristics are also applicable to DMA operations.
6. TQVWX should not be confused with TQVWX as specified for 80C51BH. On 80C152, TQVWX is measured from data valid to rising edge of $\overline{WR}$. On 80C51BH, TQVWX is measured from data valid to falling edge of $\overline{WR}$. See timing diagrams.
7. This value is based on the maximum allowable die temperature and the thermal resistance of the package.

MAX I_{CC} (ACTIVE) = (2.24 × FREQ) + 4.16 (Note 4)
MAX I_{CC} (IDLE) = (0.8 × FREQ) + 2.2 (Note 4)
where FREQ is the external oscillator Frequency in Megahertz and I_{CC} is in Milliamps



270188-17

Figure 5. I_{CC} vs Frequency

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

A: Address.
C: Clock
D: Input data.
H: Logic level HIGH.
I: Instruction (program memory contents).
L: Logic level LOW, or ALE.
P: $\overline{PSEN}$.
Q: Output data.
R: $\overline{READ}$ signal.
T: Time.
V: Valid.
W: $\overline{WRITE}$ signal.
X: No longer a valid logic level.
Z: Float.

For example,

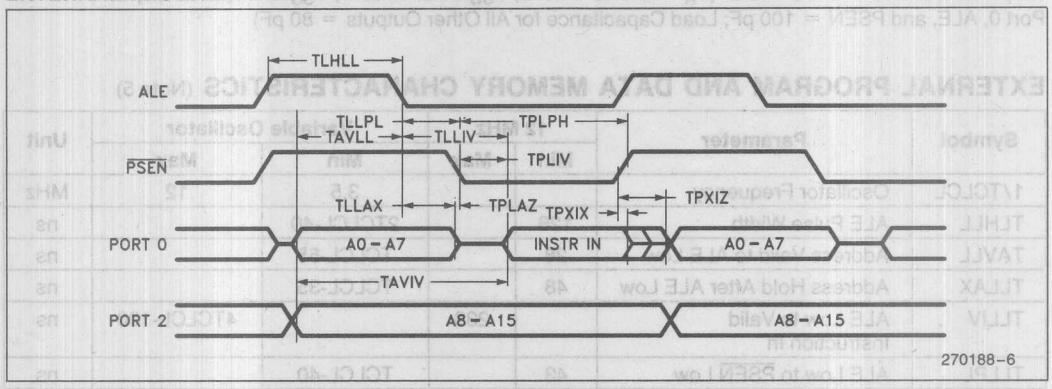
TAVLL = Time for Address Valid to ALE Low.
TLLPL = Time for ALE Low to $\overline{PSEN}$ Low.

A.C. CHARACTERISTICS (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF; Load Capacitance for All Other Outputs = 80 pF)

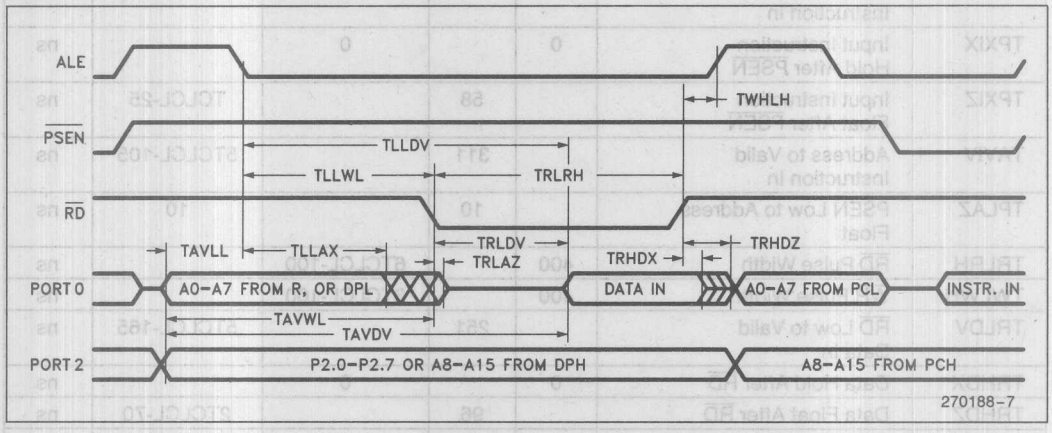
EXTERNAL PROGRAM AND DATA MEMORY CHARACTERISTICS (Note 5)

Symbol	Parameter	12 MHz		Variable Oscillator		Unit
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12	MHz
TLHLL	ALE Pulse Width	126		2TCLCL-40		ns
TAVLL	Address Valid to ALE Low	28		TCLCL-55		ns
TLLAX	Address Hold After ALE Low	48		TCLCL-35		ns
TLLIV	ALE Low to Valid Instruction In		233		4TCLCL-100	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	43		TCLCL-40		ns
TPLPH	$\overline{PSEN}$ Pulse Width	205		3TCLCL-45		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instruction In		145		3TCLCL-105	ns
TPXIX	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instruction Float After $\overline{PSEN}$		58		TCLCL-25	ns
TAVIV	Address to Valid Instruction In		311		5TCLCL-105	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		10		10	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL-100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL-100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		251		5TCLCL-165	ns
TRHDX	Data Hold After $\overline{RD}$	0		0		ns
TRHDZ	Data Float After $\overline{RD}$		96		2TCLCL-70	ns
TLLDV	ALE Low to Valid Data In		516		8TCLCL-150	ns
TAVDV	Address to Valid Data In		585		9TCLCL-165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL-50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL-130		ns
TQVWX (Note 6)	Data Valid to $\overline{WR}$ Transition	333		6TCLCL-167		ns
TWHQX	Data Hold After $\overline{WR}$	33		TCLCL-50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	TCLCL-40	TCLCL+40	ns

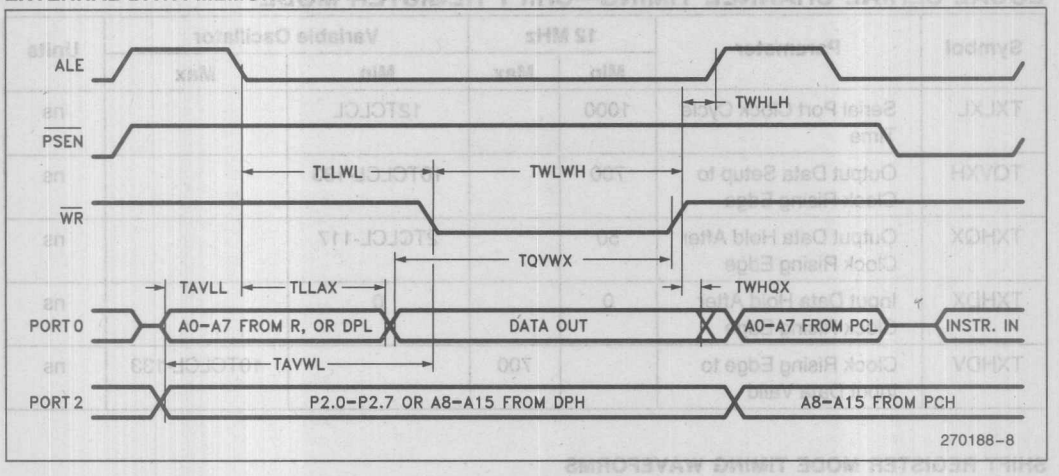
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY READ CYCLE



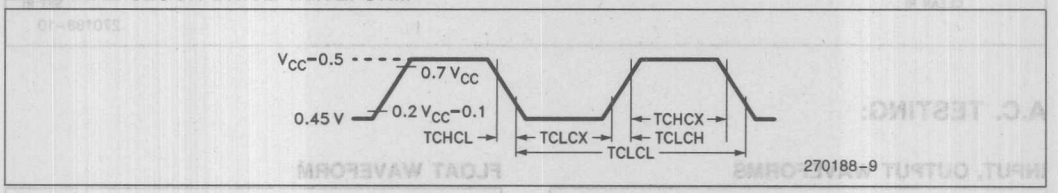
EXTERNAL DATA MEMORY WRITE CYCLE



EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

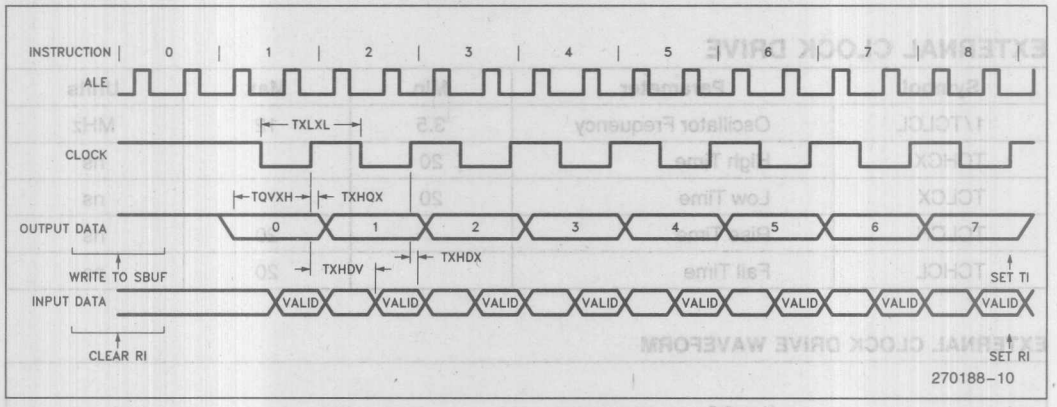
EXTERNAL CLOCK DRIVE WAVEFORM



LOCAL SERIAL CHANNEL TIMING—SHIFT REGISTER MODE

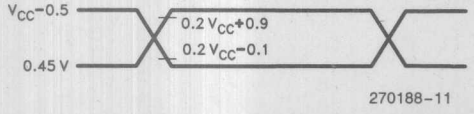
Symbol	Parameter	12 MHz		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1000		12TCLCL		ns
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL-133		ns
TXHQX	Output Data Hold After Clock Rising Edge	50		2TCLCL-117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL-133	ns

SHIFT REGISTER MODE TIMING WAVEFORMS



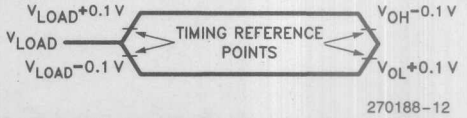
A.C. TESTING:

INPUT, OUTPUT WAVEFORMS



AC Inputs During Testing are Driven at $V_{CC}-0.5$ for a Logic "1" and 0.45V for a Logic "0". Timing Measurements are made at V_{IH} Min for a Logic "1" and V_{IL} Max for a Logic "0".

FLOAT WAVEFORM



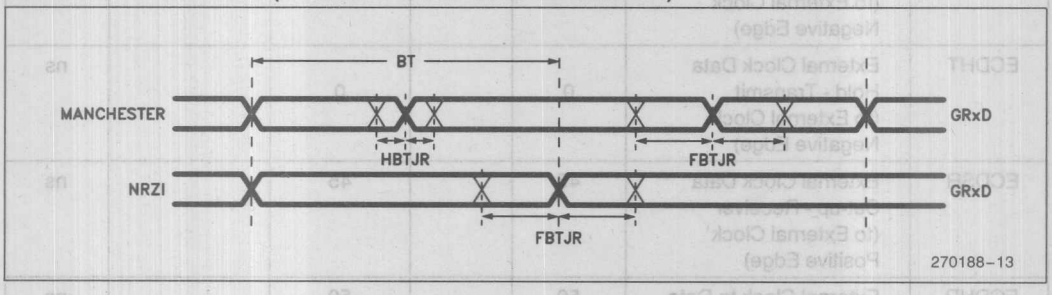
For Timing Purposes a Port Pin is no Longer Floating when a 100 mV change from Load Voltage Occurs, and Begins to Float when a 100 mV change from the Loaded V_{OH}/V_{OL} Level occurs $I_{OL}/I_{OH} \geq \pm 20$ mA

GLOBAL SERIAL PORT TIMINGS—Internal Baud Rate Generator

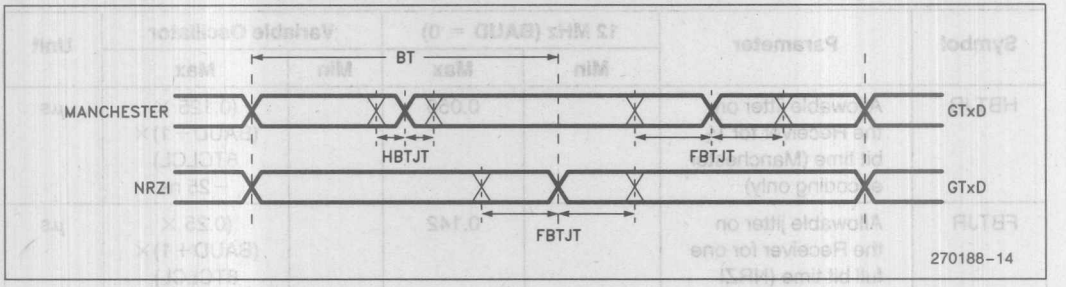
Symbol	Parameter	12 MHz (BAUD = 0)		Variable Oscillator		Unit
		Min	Max	Min	Max	
HBTJR	Allowable jitter on the Receiver for ½ bit time (Manchester encoding only)		0.058		(0.125 × (BAUD + 1) × 8TCLCL) − 25 ns	μs
FBTJR	Allowable jitter on the Receiver for one full bit time (NRZI and Manchester)		0.142		(0.25 × (BAUD + 1) × 8TCLCL) − 25 ns	μs
HBTJT	Jitter of data from Transmitter for ½ bit time (Manchester encoding only)		±35		±35	ns
FBTJT	Jitter of data from Transmitter for one full bit time (NRZI and Manchester)		±70		±70	ns
DRTR	Data rise time for Receiver (Note 8)		20		20	ns
DFTR	Data fall time for Receiver (Note 9)		20		20	ns

NOTES:
8. Same as TCLCH, use External Clock Drive Waveform.
9. Same as TCHCL, use External Clock Drive Waveform.

GSC RECEIVER TIMINGS (INTERNAL BAUD RATE GENERATOR)



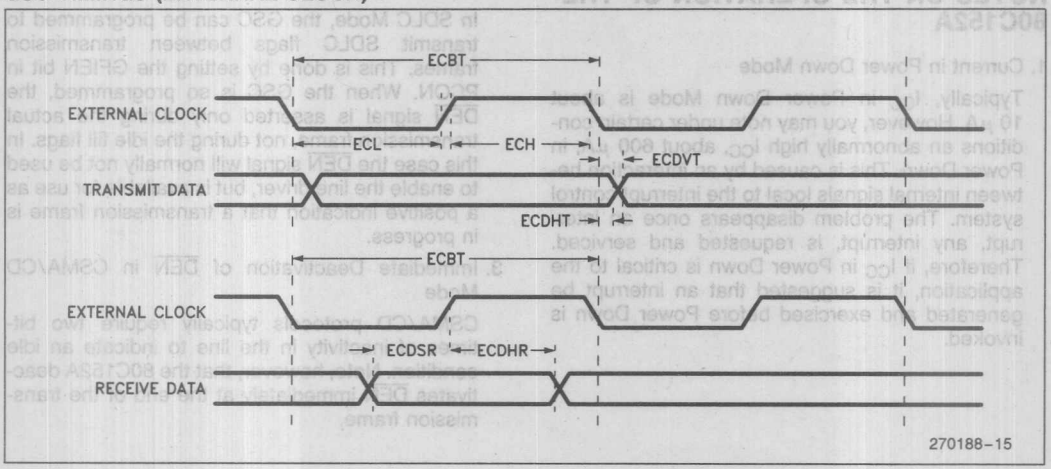
GSC TRANSMIT TIMINGS (INTERNAL BAUD RATE GENERATOR)



GLOBAL SERIAL PORT TIMINGS—External Clock

Symbol	Parameter	12 MHz		Variable Oscillator		Unit
		Min	Max	Min	Max	
1/ECBT	GSC Frequency with an External Clock	0.009	2.4	0.009	1/5TCLCL	MHz
ECH	External Clock High	197		2TCLCL + 30 ns		ns
ECL	External Clock Low	197		2TCLCL + 30 ns		ns
ECRT	External Clock Rise Time (Note 8)		20		20	ns
ECFT	External Clock Fall Time (Note 9)		20		20	ns
ECDVT	External Clock to Data Valid Out - Transmit (to External Clock Negative Edge)		150		150	ns
ECDHT	External Clock Data Hold - Transmit (to External Clock Negative Edge)	0		0		ns
ECDSR	External Clock Data Set-up - Receiver (to External Clock Positive Edge)	45		45		ns
ECDHR	External Clock to Data Hold - Receiver (to External Clock Positive Edge)	50		50		ns

GSC TIMINGS (EXTERNAL CLOCK)



NOTES ON THE OPERATION OF THE 80C152A

1. Current in Power Down Mode

   Typically, I_{CC} in Power Down Mode is about 10 μA. However, you may note under certain conditions an abnormally high I_{CC}, about 600 μA, in Power Down. This is caused by an interaction between internal signals local to the interrupt control system. The problem disappears once an interrupt, any interrupt, is requested and serviced. Therefore, if I_{CC} in Power Down is critical to the application, it is suggested that an interrupt be generated and exercised before Power Down is invoked.

2. SDLC Flags While Idling

   In SDLC Mode, the GSC can be programmed to transmit SDLC flags between transmission frames. This is done by setting the GFIEN bit in PCON. When the GSC is so programmed, the $\overline{DEN}$ signal is asserted only during the actual transmission frame, not during the idle fill flags. In this case the $\overline{DEN}$ signal will normally not be used to enable the line driver, but is available for use as a positive indication that a transmission frame is in progress.

3. Immediate Deactivation of $\overline{DEN}$ in CSMA/CD Mode

   CSMA/CD protocols typically require two bittimes of inactivity in the line to indicate an idle condition. Note, however, that the 80C152A deactivates $\overline{DEN}$ immediately at the end of the transmission frame.

83C152JA/83C152JA-1 UNIVERSAL COMMUNICATION CONTROLLER 8-BIT MICROCOMPUTER WITH FACTORY MASKED PROGRAMMABLE ROM

80C152JA/80C152JA-1 UNIVERSAL COMMUNICATION CONTROLLER 8-BIT MICROCOMPUTER

80C152JB/80C152JB-1 UNIVERSAL COMMUNICATION CONTROLLER 8-BIT MICROCOMPUTER WITH EXTENDED I/O

- Superset of 80C51 Architecture
- Multi-Protocol Serial Communication I/O Port (2.048 Mbps/2.4 Mbps Max)
  - SDLC
  - HDLC
  - CSMA/CD
  - User Definable Protocols
- Full Duplex/Half Duplex
- MCS®-51 Compatible UART
- 16.5 MHz Maximum Clock Frequency
- Multiple Power Conservation Modes
- 64KB Program Memory Addressing
- 64KB Data Memory Addressing
- 256 Bytes On-Chip RAM
- Dual On-Chip DMA Channels
- Hold/Hold Acknowledge
- Two General Purpose Timer/Counters
- 56 Special Function Registers
- 11 Interrupt Sources
- Available in 48 Pin Dual-in-Line Package and 68 Pin Surface Mount PLCC Package

  (See Packaging Spec. Order #231369)

The 80C152, which is based on the MCS®-51 CPU, is a highly integrated single-chip 8-bit microcontroller designed for cost-sensitive, high-speed, serial communications. It is well suited for implementing Integrated Services Digital Networks (ISDN), emerging Local Area Networks, and user defined serial backplane applications. In addition to the multi-protocol communication capability, the 80C152 offers traditional microcontroller features for peripheral I/O interface and control.

Silicon implementations are much more cost effective than multi-wire cables found in board level parallel-to-serial and serial-to-parallel converters. The 83C152 contains, in silicon, all the features needed for the serial-to-parallel conversion. Other 83C152 benefits include: 1) better noise immunity through differential signaling or fiber optic connections, 2) data integrity utilizing the standard, designed in CRC checks, and 3) better modularity of hardware and software designs. All of these—cost, network parameter and real estate improvements—apply to 83C152 serial links between boards or systems and 83C152 serial links on a single board.

September 1987
Order Number: 270431-001

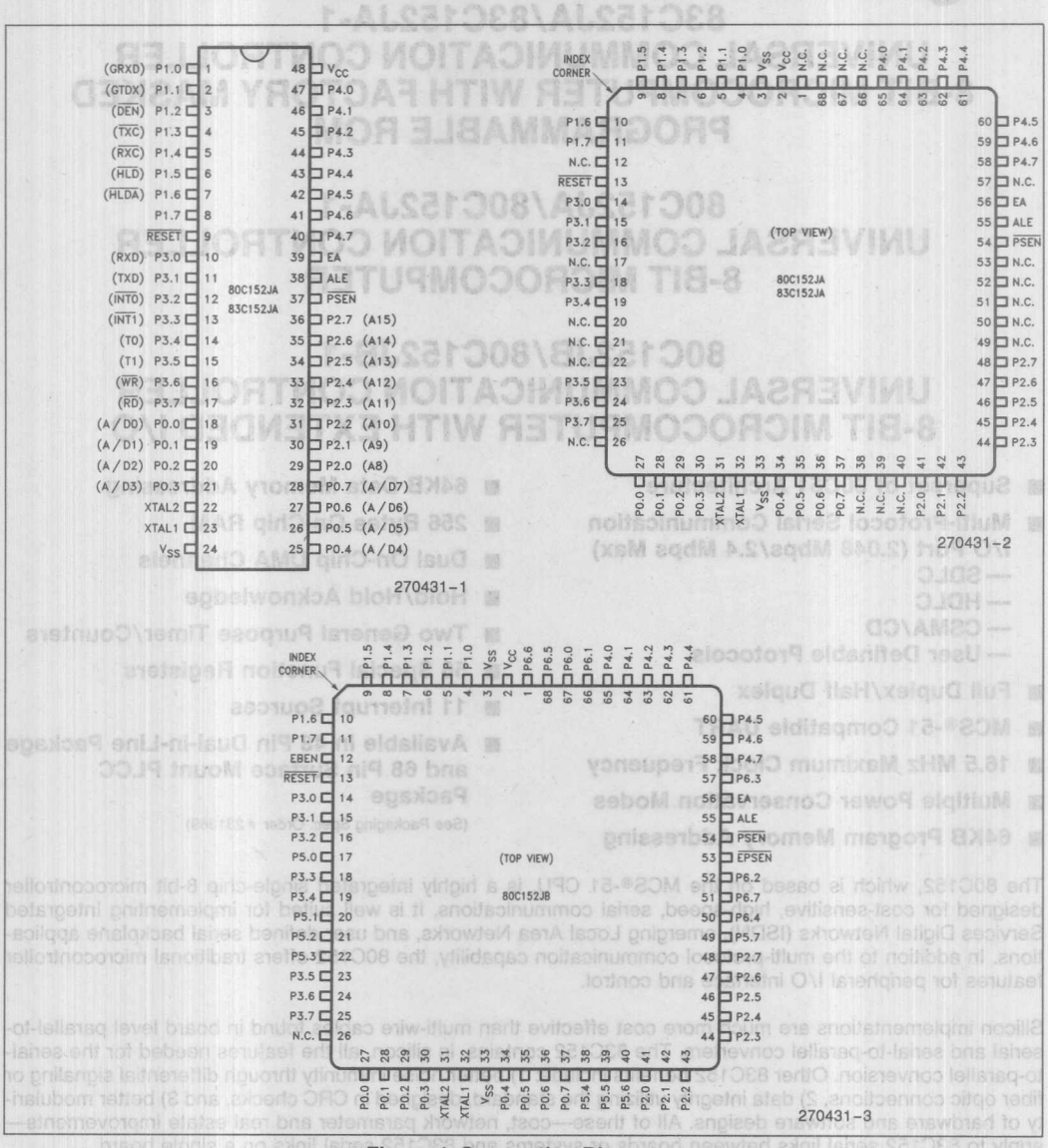
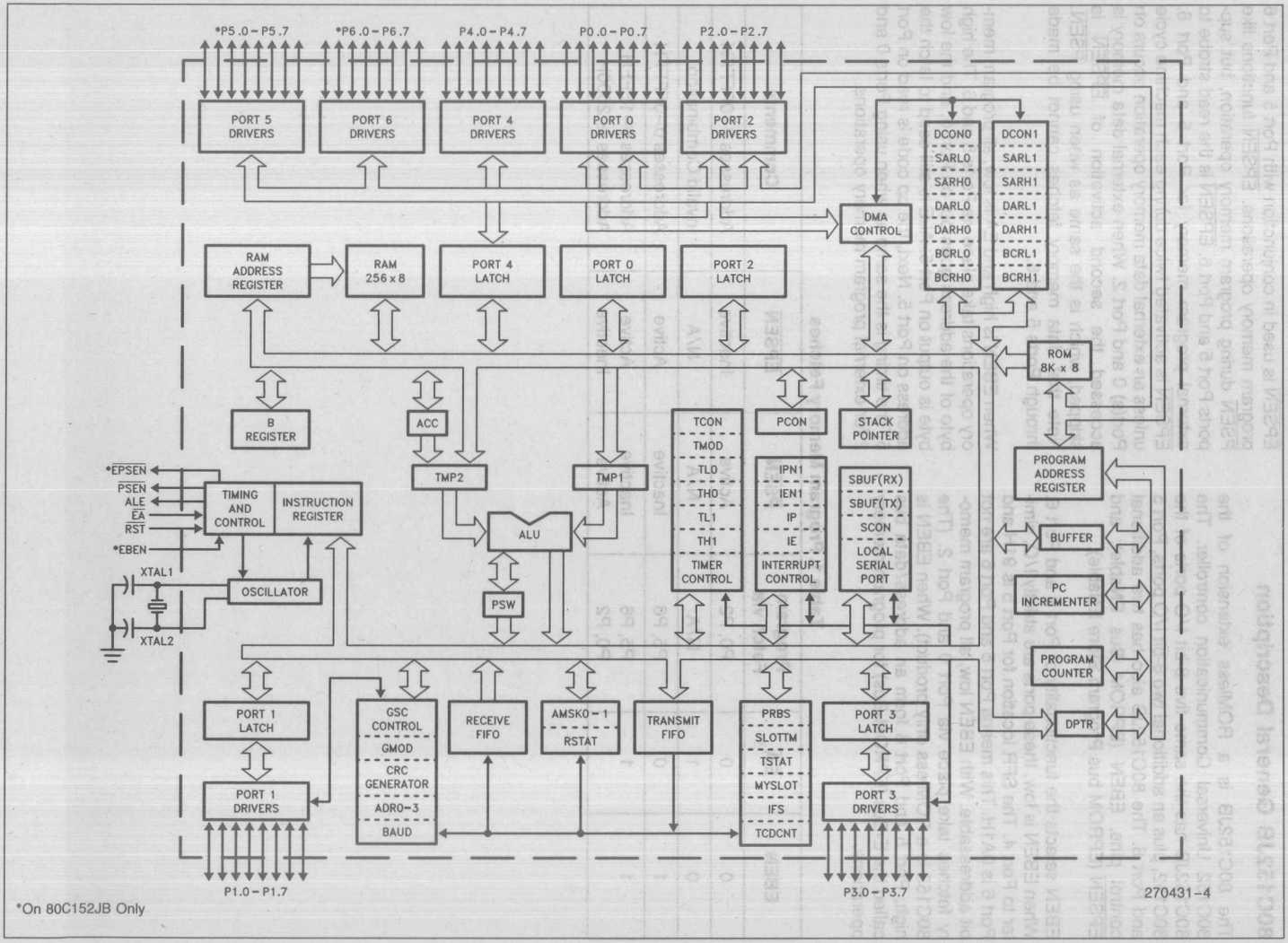


Figure 1. Connection Diagrams



*On 80C152JB Only

Figure 2. Block Diagram

80C152JB General Description

The 80C152JB is a ROMless extension of the 80C152 Universal Communication controller. The 80C152JB has the same five 8-bit I/O ports of the 80C152, plus an additional two 8-bit I/O ports, Port 5 and Port 6. The 80C152JB also has two additional control pins, EBEN (**E**PROM **B**us **EN**able), and $\overline{\text{EPSEN}}$ (**E**PROM bus **P**rogram **S**tore **EN**able).

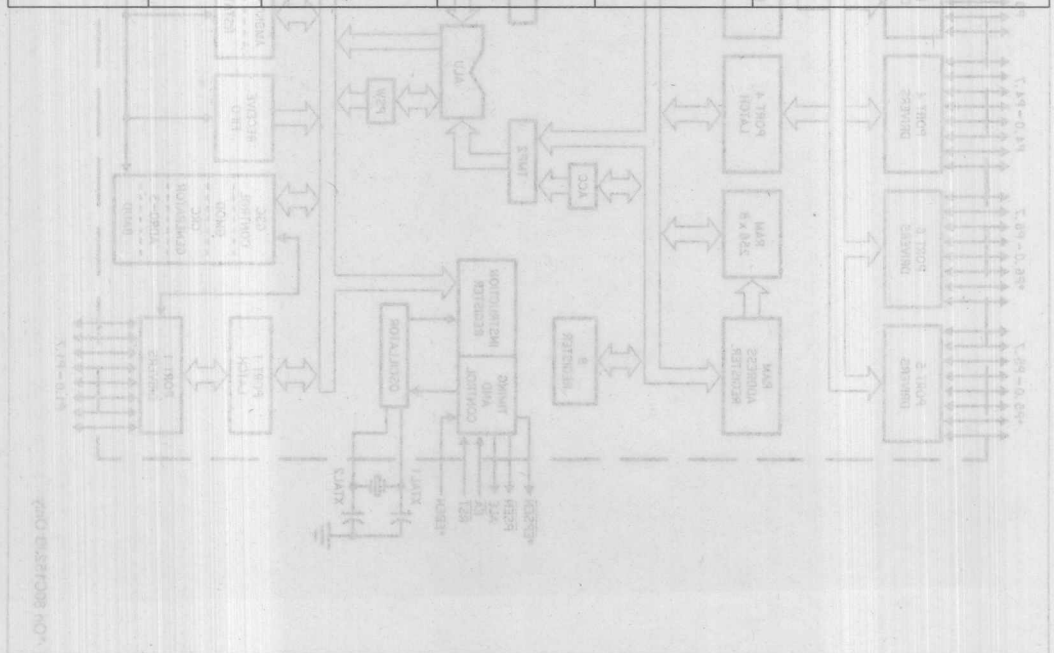
EBEN selects the functionality of Port 5 and Port 6. When EBEN is low, these ports are strictly I/O, similar to Port 4. The SFR location for Port 5 is 91H and Port 6 is 0A1H. This means Port 5 and Port 6 are not bit addressable. With EBEN low, all program memory fetches take place via Port 0 and Port 2. (The 80C152 is a ROMless only product). When EBEN is high, Port 5 and Port 6 form an address/data bus called the E-Bus (EPROM-Bus) for program memory operations.

$\overline{\text{EPSEN}}$ is used in conjunction with Port 5 and Port 6 program memory operations. $\overline{\text{EPSEN}}$ functions like $\overline{\text{PSEN}}$ during program memory operation, but supports Port 5 and Port 6. $\overline{\text{EPSEN}}$ is the read strobe to external program memory for Port 5 and Port 6. $\overline{\text{EPSEN}}$ is activated twice during each machine cycle unless an external data memory operation occurs on Port(s) 0 and Port 2. When external data memory is accessed the second activation of $\overline{\text{EPSEN}}$ is skipped, which is the same as when using $\overline{\text{PSEN}}$. Note that data memory fetches cannot be made through Ports 5 and 6.

When EBEN is high and EA is low, all program memory operations take place via Ports 5 and 6. The high byte of the address goes out on Port 6, and the low byte is output on Port 5. ALE is still used to latch the address on Port 5. Next, the op code is read on Port 5. The timing is the same as when using Ports 0 and 2 for external program memory operations.

Table 1. Program Memory Fetches

EBEN	$\overline{\text{EA}}$	Program Fetch via	$\overline{\text{PSEN}}$	$\overline{\text{EPSEN}}$	Comments
0	0	P0, P2	Active	Inactive	Addresses 0–0FFFFH
0	1	N/A	N/A	N/A	Invalid Combination
1	0	P5, P6	Inactive	Active	Addresses 0–0FFFFH
1	1	P5, P6 P0, P2	Inactive Active	Active Inactive	Addresses 0–1FFFH Addresses ≥ 2000H



Pin #		Pin Description
DIP	PLCC(1)	
48	2	V_{CC}—Supply voltage.
24	3,33(2)	V_{SS}—Circuit ground.
18-21, 25-28	27-30, 34-37	**Port 0**—Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program memory if EBEN is pulled low. During accesses to external Data Memory, Port 0 always emits the low-order address byte and serves as the multiplexed data bus. In these applications it uses strong internal pullups when emitting 1s. Port 0 also outputs the code bytes during program verification. External pullups are required during program verification.
1-8	4-11	**Port 1**—Port 1 is an 8-bit bidirectional I/O port with internal pullups. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups. Port 1 also serves the functions of various special features of the 8XC152, as listed below:

Pin	Name	Alternate Function
P1.0	GRXD	GSC data input pin
P1.1	GTXD	GSC data output pin
P1.2	$\overline{DEN}$	GSC enable signal for an external driver
P1.3	$\overline{TXC}$	GSC input pin for external transmit clock
P1.4	$\overline{RXC}$	GSC input pin for external receive clock
P1.5	$\overline{HLD}$	DMA hold input/output
P1.6	$\overline{HLDA}$	DMA hold acknowledge input/output

29-36	41-48	**Port 2**—Port 2 is an 8-bit bidirectional I/O port with internal pullups. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups. Port 2 emits the high-order address byte during fetches from external Program Memory if EBEN is pulled low. During accesses to external Data Memory that use 16-bit addresses (MOVX @ DPTR and DMA operations), Port 2 emits the high-order address byte. In these applications it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @ Ri), Port 2 emits the contents of the P2 Special Function Register. Port 2 also receives the high-order address bits during program verification.
10- 17	14-16, 18, 19, 23-25	**Port 3**—Port 3 is an 8-bit bidirectional I/O port with internal pullups. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the pullups. Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Pin	Name	Alternate Function
P3.0	RXD	Serial input line
P3.1	TXD	Serial output line
P3.2	$\overline{INT0}$	External Interrupt 0
P3.3	$\overline{INT1}$	External Interrupt 1
P3.4	T0	Timer 0 external input
P3.5	T1	Timer 1 external input
P3.6	$\overline{WR}$	External Data Memory Write strobe
P3.7	$\overline{RD}$	External Data Memory Read strobe

NOTES:
1. N.C. pins on PLCC package may be connected to internal die and should not be used in customer applications.
2. It is recommended that both Pin 3 and Pin 33 be grounded for PLCC devices.

Pin Description (Continued)

Pin #		Pin Description
47-40	65-58	**Port 4**—Port 4 is an 8-bit bidirectional I/O port with internal pullups. Port 4 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 4 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups. In addition, Port 4 also receives the low-order address bytes during program verification.
9	13	**$\overline{RST}$**—Reset input. A logic low on this pin for three machine cycles while the oscillator is running resets the device. An internal pullup resistor permits a power-on reset to be generated using only an external capacitor to V_{SS}. Although the GSC recognizes the reset after three machine cycles, data may continue to be transmitted for up to 4 machine cycles after Reset is first applied.
38	55	**ALE**—Address Latch Enable output signal for latching the low byte of the address during accesses to external memory. In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory. While in Reset, ALE remains at a constant high level.
37	54	**$\overline{PSEN}$**—Program Store Enable is the Read strobe to External Program Memory. When the 8XC152 is executing from external program memory, $\overline{PSEN}$ is active (low). When the device is executing code from External Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to External Data Memory. While in Reset, $\overline{PSEN}$ remains at a constant high level.
39	56	**$\overline{EA}$**—External Access enable. $\overline{EA}$ must be externally pulled low in order to enable the 8XC152 to fetch code from External Program Memory locations 0000H to 0FFFH. $\overline{EA}$ must be connected to V_{CC} for internal program execution.
23	32	**XTAL1**—Input to the inverting oscillator amplifier and input to the internal clock generating circuits.
22	31	**XTAL2**—Output from the inverting oscillator amplifier.
N/A	17, 20 21, 22 38, 39 40, 49	**Port 5**—Port 5 is an 8-bit bidirectional I/O port with internal pullups. Port 5 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 5 pins that are externally being pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups. Port 5 is also the multiplexed low-order address and data bus during accesses to external program memory if EBEN is pulled high. In this application it uses strong pullups when emitting 1s.
N/A	67, 66 52, 57 50, 68 1, 51	**Port 6**—Port 6 is an 8-bit bidirectional I/O port with internal pullups. Port 6 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 6 pins that are externally pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups. Port 6 emits the high-order address byte during fetches from external Program Memory if EBEN is pulled high. In this application it uses strong pullups when emitting 1s.
N/A	12	**EBEN**—E-Bus Enable input that designates whether program memory fetches take place via Ports 0 and 2 or Ports 5 and 6. Table 1 shows how the ports are used in conjunction with EBEN.
	53	**$\overline{EPSEN}$**—E-bus Program Store Enable is the Read strobe to external program memory when EBEN is high. Table 2 shows when $\overline{EPSEN}$ is used relative to $\overline{PSEN}$ depending on the status of EBEN and $\overline{EA}$.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3.

To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left unconnected, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

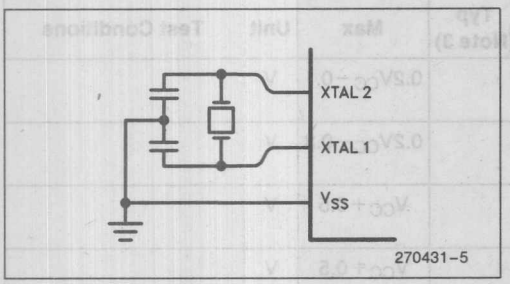


Figure 3. Using the On-Chip Oscillator

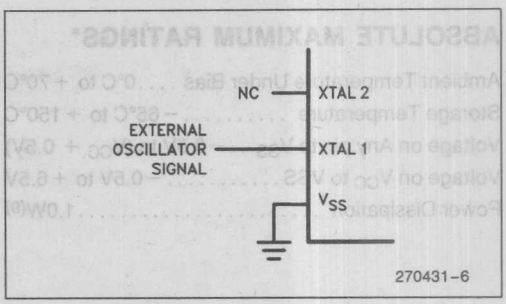


Figure 4. External Clock Drive

IDLE MODE

In Idle Mode, the CPU puts itself to sleep while most of the on-chip peripherals remain active. The major peripherals that do not remain active during Idle, are the DMA channels. The Idle Mode is invoked by software. The content of the on-chip RAM and all the Special Function Registers remain unchanged during this mode. The Idle Mode can be terminated by any enabled interrupt or by a hardware reset.

POWER DOWN MODE

In Power Down Mode, the oscillator is stopped and all on-chip functions cease except that the on-chip RAM contents are maintained. The mode Power Down is invoked by software. The Power Down Mode can be terminated only by a hardware reset.

Table 2. Status of the External Pins During Idle and Power Down Modes

80C152JA/83C152JA

Mode	Program Memory	ALE	$\overline{PSEN}$	Port 0	Port 1	Port 2	Port 3	Port 4
Idle	Internal	1	1	Data	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data	Data
Power Down	Internal	0	0	Data	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data	Data

80C152JB

Mode	Instruction Bus	ALE	$\overline{PSEN}$	$\overline{EPSEN}$	Port 0	Port 1	Port 2	Port 3	Port 4	Port 5	Port 6
Idle	P0, P2	1	1	1	Float	Data	Address	Data	Data	0FFH	0FFH
Idle	P5, P6	1	1	1	Data	Data	Data	Data	Data	0FFH	Address
Power Down	P0, P2	0	0	1	Float	Data	Data	Data	Data	0FFH	0FFH
Power Down	P5, P6	0	1	0	Data	Data	Data	Data	Data	0FFH	0FFH

NOTE:
For more detailed information on the reduced power modes refer to the Embedded Controller Handbook, and Application Note AP-252, "Designing with the 80C51BH."

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any pin to V_{SS} . . −0.5V to (V_{CC} + 0.5V)
Voltage on V_{CC} to VSS . . . . . −0.5V to +6.5V
Power Dissipation . . . . . 1.0W(9)

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V)

Symbol	Parameter	Min	Typ (Note 3)	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage (All Except $\overline{EA}$, EBEN)	−0.5		$0.2V_{CC}-0.1$	V	
V_{IL1}	Input Low Voltage ($\overline{EA}$, EBEN)	−0.5		$0.2V_{CC}-0.3$	V	
V_{IH}	Input High Voltage (Except XTAL1, $\overline{RST}$)	$0.2V_{CC}+0.9$		$V_{CC}+0.5$	V	
V_{IH1}	Input High Voltage (XTAL1, $\overline{RST}$)	$0.7V_{CC}$		$V_{CC}+0.5$	V	
V_{OL}	Output Low Voltage (Ports 1, 2, 3, 4, 5, 6)			0.45	V	I_{OL} = 1.6 mA (Note 4)
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$, $\overline{EPSEN}$)			0.45	V	I_{OL} = 3.2 mA (Note 4)
V_{OH}	Output High Voltage (Ports 1, 2, 3, 4, 5, 6 COMM9 ALE, $\overline{PSEN}$, $\overline{EPSEN}$)	2.4			V	I_{OH} = −60 μA V_{CC} = 5V ±10%
		$0.9V_{CC}$			V	I_{OH} = −10 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4			V	I_{OH} = −400 μA V_{CC} = 5V ±10%
		$0.9V_{CC}$			V	I_{OH} = −40 μA (Note 5)
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3, 4, 5, 6)			−50	μA	V_{IN} = 0.45V
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, 3, 4, 5, 6)			−650	μA	V_{IN} = 2V
I_{LI}	Input Leakage (Port 0, $\overline{EA}$)			±10	μA	$0.45 < V_{IN} < V_{CC}$
RRST	Reset Pullup Resistor	40			kΩ	
I_{IH}	Logical 1 Input Current (EBEN)			+60	μA	
I_{CC}	Power Supply Current :					
	Active (16.5 MHz)		31	41.1	mA	(Note 6)
	Idle (16.5 MHz)		8	15.4	mA	(Note 6)
	Power Down Mode		10		μA	V_{CC} = 2.0V to 5.5V

NOTES:
3. "Typicals" are based on samples taken from early manufacturing lots and are not guaranteed. The measurements were made with V_{CC} = 5V at room temperature.
4. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
5. Capacitive loading on Ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
6. I_{CC} is measured with all output pins disconnected; XTAL1 driven with TCLCH, TCHCL = 5 ns, $V_{IL} = V_{SS} + 0.5V$, $V_{IH} = V_{CC} - 0.5V$; XTAL2 N.C.; Port 0 pins connected to V_{CC}. "Operating" current is measured with $\overline{EA}$ connected to V_{CC} and $\overline{RST}$ connected to V_{SS}. "Idle" current is measured with $\overline{EA}$ connected to V_{SS}, $\overline{RST}$ connected to V_{CC} and GSC inactive.
7. The specifications relating to external data memory characteristics are also applicable to DMA operations.
8. TQVWX should not be confused with TQVWX as specified for 80C51BH. On 80C152, TQVWX is measured from data valid to rising edge of $\overline{WR}$. On 80C51BH, TQVWX is measured from data valid to falling edge of $\overline{WR}$. See timing diagrams.
9. This value is based on the maximum allowable die temperature and the thermal resistance of the package.
10. All specifications relating to external program memory characteristics are applicable to:
    $\overline{EPSEN}$ for $\overline{PSEN}$
    Port 5 for Port 0
    Port 6 for Port 2
    when EBEN is at a Logical 1 on the 80C152JB.

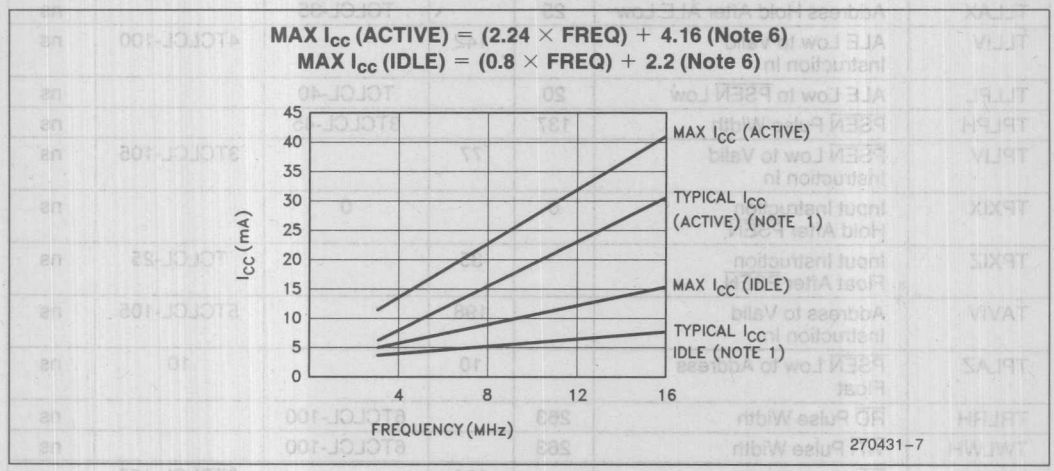


Figure 5. I_{CC} vs Frequency

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

A: Address.
C: Clock
D: Input data.
H: Logic level HIGH.
I: Instruction (program memory contents).
L: Logic level LOW, or ALE.
P: $\overline{PSEN}$.
Q: Output data.
R: $\overline{READ}$ signal.
T: Time.
V: Valid.
W: $\overline{WRITE}$ signal.
X: No longer a valid logic level.
Z: Float.

For example,

TAVLL = Time for Address Valid to ALE Low.
TLLPL = Time for ALE Low to $\overline{PSEN}$ Low.

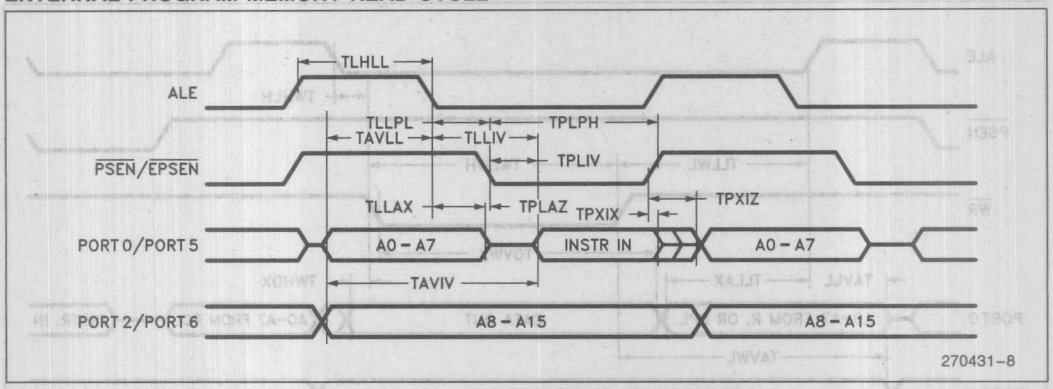
A.C. CHARACTERISTICS (T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF; Load Capacitance for All Other Outputs = 80 pF)

ADVANCE INFORMATION: SEE INTEL FOR DESIGN-IN INFORMATION

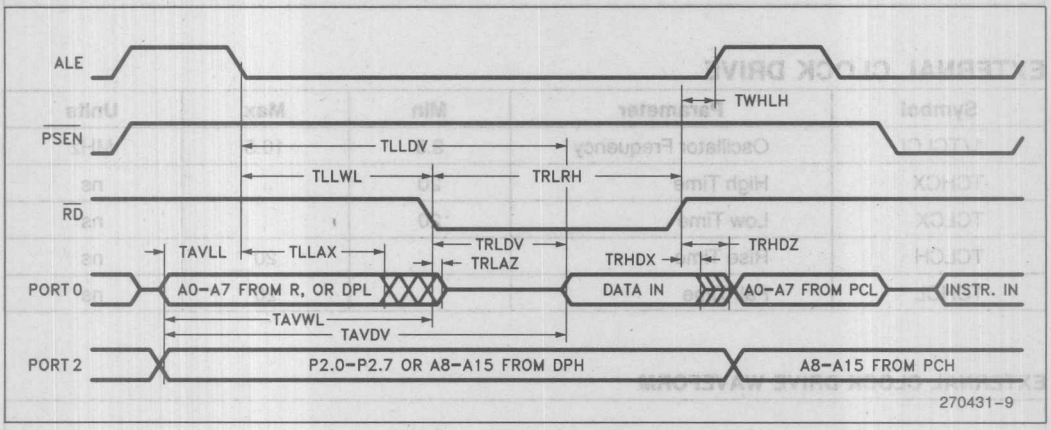
EXTERNAL PROGRAM AND DATA MEMORY CHARACTERISTICS (Note 7, 10)

Symbol	Parameter	16.5 MHz		Variable Oscillator		Unit
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency 80C152JA 83C152JA 80C152JB			3.5	12	MHz
	80C152JA-1 83C152JA-1 80C152JB-1			3.5	16.5	MHz
TLHLL	ALE Pulse Width	81		2TCLCL-40		ns
TAVLL	Address Valid to ALE Low	5		TCLCL-55		ns
TLLAX	Address Hold After ALE Low	25		TCLCL-35		ns
TLLIV	ALE Low to Valid Instruction In		142		4TCLCL-100	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	20		TCLCL-40		ns
TPLPH	$\overline{PSEN}$ Pulse Width	137		3TCLCL-45		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instruction In		77		3TCLCL-105	ns
TPXIX	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instruction Float After $\overline{PSEN}$		35		TCLCL-25	ns
TAVIV	Address to Valid Instruction In		198		5TCLCL-105	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		10		10	ns
TRLRH	$\overline{RD}$ Pulse Width	263		6TCLCL-100		ns
TWLWH	$\overline{WR}$ Pulse Width	263		6TCLCL-100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		138		5TCLCL-165	ns
TRHDX	Data Hold After $\overline{RD}$	0		0		ns
TRHDZ	Data Float After $\overline{RD}$		51		2TCLCL-70	ns
TLLDV	ALE Low to Valid Data In		335		8TCLCL-150	ns
TAVDV	Address to Valid Data In		380		9TCLCL-165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	132	232	3TCLCL-50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	112		4TCLCL-130		ns
TQVWX(8)	Data Valid to $\overline{WR}$ Transition	196		6TCLCL-167		ns
TWHQX	Data Hold After $\overline{WR}$	10		TCLCL-50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	20	100	TCLCL-40	TCLCL+40	ns

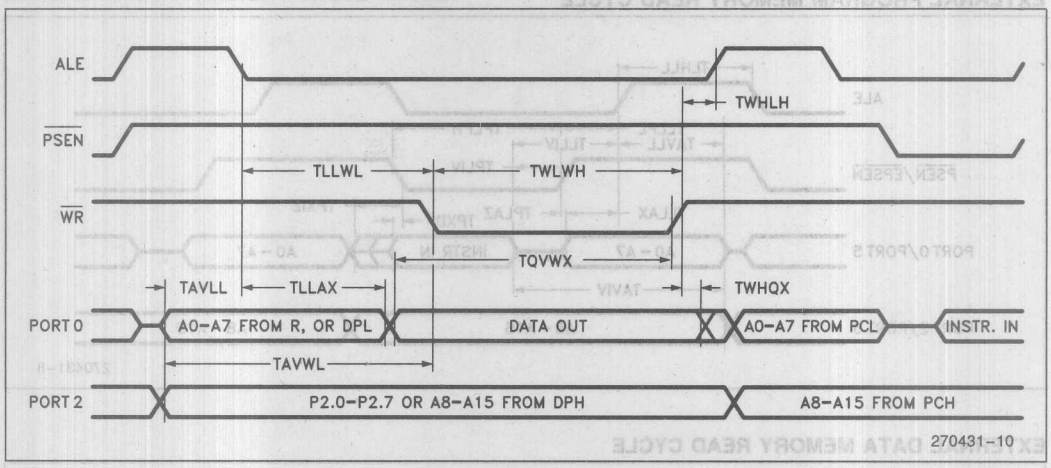
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY READ CYCLE



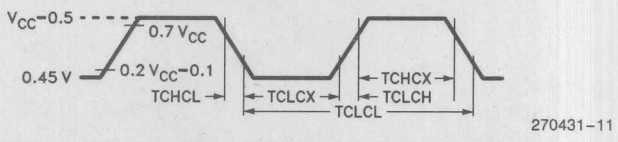
EXTERNAL DATA MEMORY WRITE CYCLE



EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	16.5	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

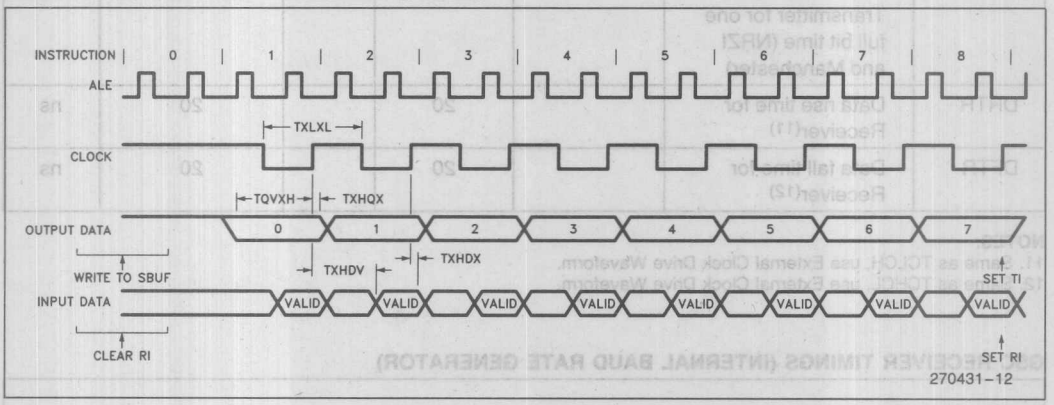
EXTERNAL CLOCK DRIVE WAVEFORM



LOCAL SERIAL CHANNEL TIMING—SHIFT REGISTER MODE

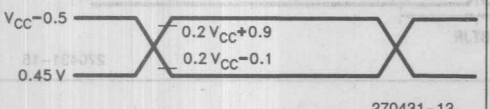
Symbol	Parameter	16.5 MHz		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	727		12TCLCL		ns
TQVXH	Output Data Setup to Clock Rising Edge	473		10TCLCL-133		ns
TXHQX	Output Data Hold After Clock Rising Edge	4		2TCLCL-117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		473		10TCLCL-133	ns

SHIFT REGISTER MODE TIMING WAVEFORMS



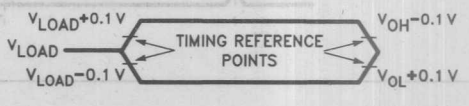
A.C. TESTING:

INPUT, OUTPUT WAVEFORMS



AC Inputs During Testing are Driven at $V_{CC}-0.5$ for a Logic "1" and 0.45V for a Logic "0". Timing Measurements are made at V_{IH} Min for a Logic "1" and V_{IL} Max for a Logic "0".

FLOAT WAVEFORM



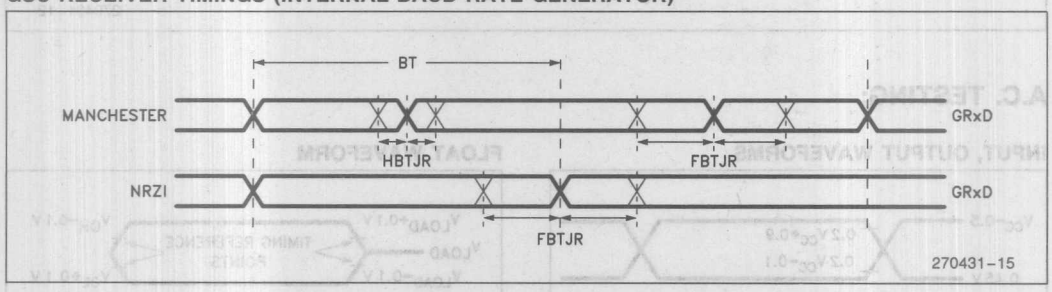
For Timing Purposes a Port Pin is no Longer Floating when a 100 mV change from Load Voltage Occurs, and Begins to Float when a 100 mV change from the Loaded V_{OH}/V_{OL} Level occurs $I_{OL}/I_{OH} \geq \pm 20$ mA.

GLOBAL SERIAL PORT TIMINGS—Internal Baud Rate Generator

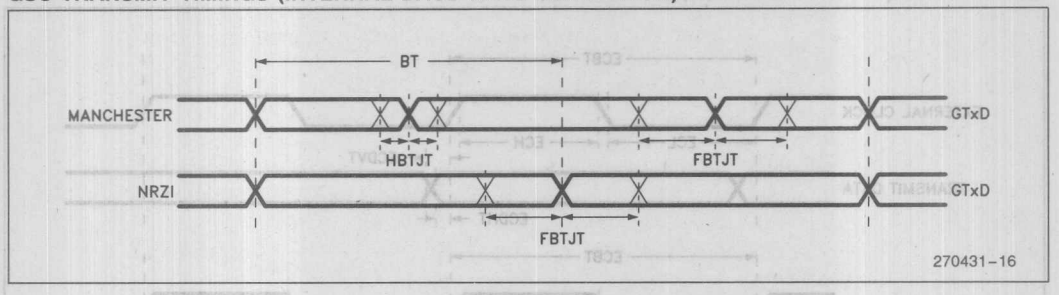
Symbol	Parameter	16.5 MHz (BAUD = 0)		Variable Oscillator		Unit
		Min	Max	Min	Max	
HBTJR	Allowable jitter on the Receiver for ½ bit time (Manchester encoding only)		0.0375		(0.125 × (BAUD+1)× 8TCLCL) −25 ns	µs
FBTJR	Allowable jitter on the Receiver for one full bit time (NRZI and Manchester)		0.10		(0.25 × (BAUD+1)× 8TCLCL) −25 ns	µs
HBTJT	Jitter of data from Transmitter for ½ bit time (Manchester encoding only)		±35		±35	ns
FBTJT	Jitter of data from Transmitter for one full bit time (NRZI and Manchester)		±70		±70	ns
DRTR	Data rise time for Receiver(11)		20		20	ns
DFTR	Data fall time for Receiver(12)		20		20	ns

NOTES:
11. Same as TCLCH, use External Clock Drive Waveform.
12. Same as TCHCL, use External Clock Drive Waveform.

GSC RECEIVER TIMINGS (INTERNAL BAUD RATE GENERATOR)



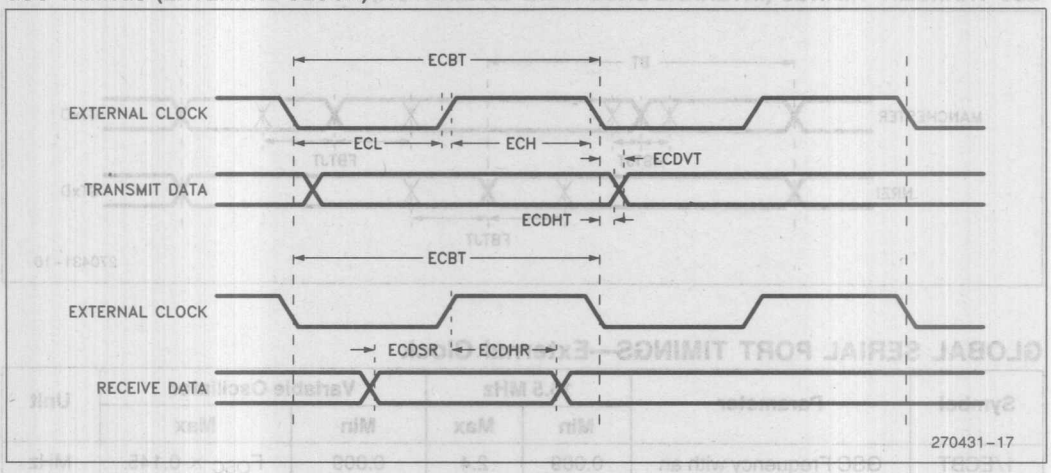
GSC TRANSMIT TIMINGS (INTERNAL BAUD RATE GENERATOR)



GLOBAL SERIAL PORT TIMINGS—External Clock

Symbol	Parameter	16.5 MHz		Variable Oscillator		Unit
		Min	Max	Min	Max	
1/ECBT	GSC Frequency with an External Clock	0.009	2.4	0.009	$F_{OSC} \times 0.145$	MHz
ECH	External Clock High	155		2TCLCL + 30 ns		ns
ECL	External Clock Low	155		2TCLCL + 30 ns		ns
ECRT	External Clock Rise Time(11)		20		20	ns
ECFT	External Clock Fall Time(12)		20		20	ns
ECDVT	External Clock to Data Valid Out - Transmit (to External Clock Negative Edge)		150		150	ns
ECDHT	External Clock Data Hold - Transmit (to External Clock Negative Edge)	0		0		ns
ECDSR	External Clock Data Set-up - Receiver (to External Clock Positive Edge)	45		45		ns
ECDHR	External Clock to Data Hold - Receiver (to External Clock Positive Edge)	50		50		ns

GSC TIMINGS (EXTERNAL CLOCK)



27C64/87C64
64K (8K x 8) CHMOS PRODUCTION AND UV ERASABLE PROMS

- **CHMOS Microcontroller and Microprocessor Compatible**
  - **87C64-Integrated Address Latch**
  - **Universal 28 Pin Memory Site, 2-line Control**
- **Low Power Consumption**
  - **100 μA Maximum Standby Current**
- **Noise Immunity Features**
  - **$\pm$10% V_{CC} Tolerance**
  - **Maximum Latch-up Immunity Through EPI Processing**
- **High Performance Speeds**
  - **150 ns Maximum Access Time**
- **New Quick-Pulse Programming™ Algorithm (1 second programming)**
- **Available in 28-Pin Cerdip and Plastic DIP Package and 32-Lead PLCC Package.**

(See Packaging Spec, Order #231369)

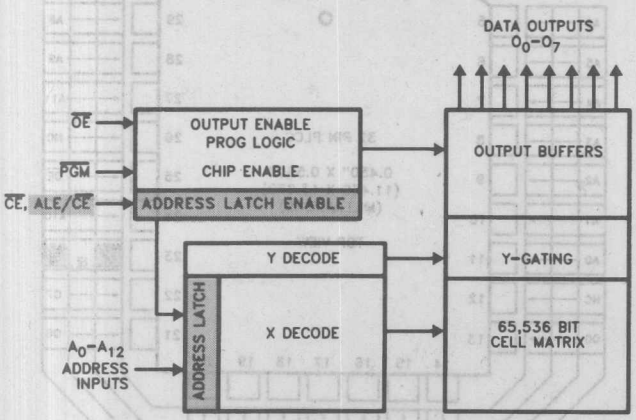
Intel's 27C64 and 87C64 CHMOS EPROMs are 64K bit 5V only memories organized as 8192 words of 8 bits. They employ advanced CHMOS*II-E circuitry for systems requiring low power, high performance speeds, and immunity to noise. The 87C64 has been optimized for multiplexed bus microcontroller and microprocessor compatibility while the 27C64 has a non-multiplexed addressing interface and is plug compatible with the standard Intel 2764A (HMOS II-E).

The 27C64 and 87C64 are offered in both a ceramic DIP, Plastic DIP, and Plastic Leaded Chip Carrier (PLCC) Packages. Cerdip packages provide flexibility in prototyping and R&D environments, whereas Plastic DIP and PLCC EPROMs provide optimum cost effectiveness in production environments. A new Quick-Pulse Programming™ Algorithm is employed which can speed up programming by as much as one hundred times.

The 87C64 incorporates an address latch on the address pins to minimize chip count in multiplexed bus systems. Designers can eliminate an external address latch by tieing address and data pins of the 87C64 directy to the processor's multiplexed address/data pins. On the falling edge of the ALE input (ALE/$\overline{CE}$), address information at the address inputs (A_0–A_{12}) of the 87C64 is latched internally. The address inputs are then ignored as data information is passed on the same bus.

The highest degree of protection against latch-up is achieved through Intel's unique EPI processing. Prevention of latch-up is provided for stresses up to 100 mA on address and data pins from -1V to $V_{CC} + 1$V.

*HMOS and CHMOS are patented processes of Intel Corporation.



Shaded Areas represent the 87C64 version

290000-1

Figure 1. Block Diagram

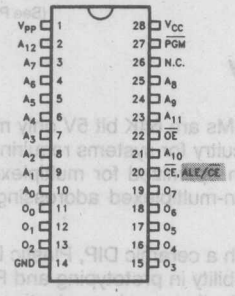
October 1987
Order Number: 290000-007

Pin Names

A_0-A_{12}	ADDRESSES
O_0-O_7	OUTPUTS
$\overline{OE}$	OUTPUT ENABLE
$\overline{CE}$	CHIP ENABLE
ALE/$\overline{CE}$	ADDRESS LATCH ENABLE /CHIP ENABLE
$\overline{PGM}$	PROGRAM STROBE
N.C.	NO CONNECT
D.U.	DON'T USE

27C64/87C64
P27C64/P87C64

27256	27128	2732A	2716
V_{PP}	V_{PP}		
A_{12}	A_{12}		
A_7	A_7	A_7	A_7
A_6	A_6	A_6	A_6
A_5	A_5	A_5	A_5
A_4	A_4	A_4	A_4
A_3	A_3	A_3	A_3
A_2	A_2	A_2	A_2
A_1	A_1	A_1	A_1
A_0	A_0	A_0	A_0
O_0	O_0	O_0	O_0
O_1	O_1	O_1	O_1
O_2	O_2	O_2	O_2
Gnd	Gnd	Gnd	Gnd

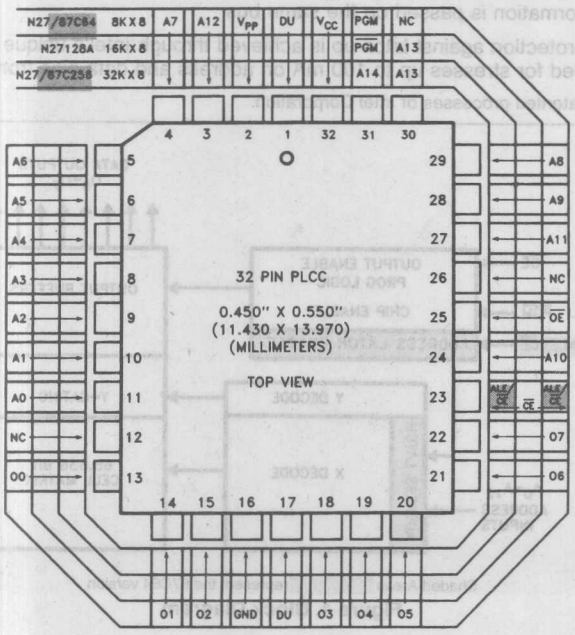


290000-2

2716	2732A	27128	27256
		V_{CC}	V_{CC}
		$\overline{PGM}$	A_{14}
V_{CC}	V_{CC}	A_{13}	A_{13}
A_8	A_8	A_8	A_8
A_9	A_9	A_9	A_9
V_{PP}	A_{11}	A_{11}	A_{11}
$\overline{OE}$	$\overline{OE}/V_{PP}$	$\overline{OE}$	$\overline{OE}$
A_{10}	A_{10}	A_{10}	A_{10}
$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$
O_7	O_7	O_7	O_7
O_6	O_6	O_6	O_6
O_5	O_5	O_5	O_5
O_4	O_4	O_4	O_4
O_3	O_3	O_3	O_3

NOTE:
Intel "Universal Site" Compatible EPROM Pin Configurations are shown in the adjacent blocks to 27C64 Pins.
Shaded Areas represent the 87C64 version

Figure 2. Pin Configuration



290000-11

Figure 3. PLCC(N) Lead Configuration

Extended Temperature (Express) EPROMs

The Intel EXPRESS EPROM family is a series of electrically programmable read only memories which have received additional processing to enhance product characteristics. EXPRESS processing is available for several densities of EPROM, allowing the choice of appropriate memory size to match system applications.

EXPRESS EPROM products are available with 168 ±8 hour, 125°C dynamic burn-in using Intel's standard bias configuration. This process exceeds or meets most industry specifications of burn-in. The standard EXPRESS EPROM operating temperature range is 0°C to 70°C. Extended operating temperature range (−40°C to +85°C) EXPRESS products are available along with automotive temperature range (−40°C to +125°C) products. Like all Intel EPROMs, the EXPRESS EPROM family is inspected to 0.1% electrical AQL. This may allow the user to reduce or eliminate incoming inspection testing.

EXPRESS EPROM Product Family

PRODUCT DEFINITIONS

Type	Operating Temperature (°C)	Burn-in 125°C (hr)
Q	0 to +70	168 ±8
T	−40 to +85	NONE
L	−40 to +85	168 ±8
A	−40 to +125	NONE
B	−40 to +125	168 ±8

EXPRESS Options

27C64/87C64 Versions

Packaging Options			
Speed Versions	**Cerdip**	**PLCC**	**Plastic DIP**
−1	T, L, Q	T	T
−15	T, L, Q	T	T
−2	T, L, Q, A, B	T, A	T, A
−20	T, L, Q, A	T	T
−STD	T, L, Q, A, B	T, A	T, A
−25	T, L, Q, A	T	T
−3	T, L, Q, A, B	T, A	T, A
−30	T, L, Q, A	T	T

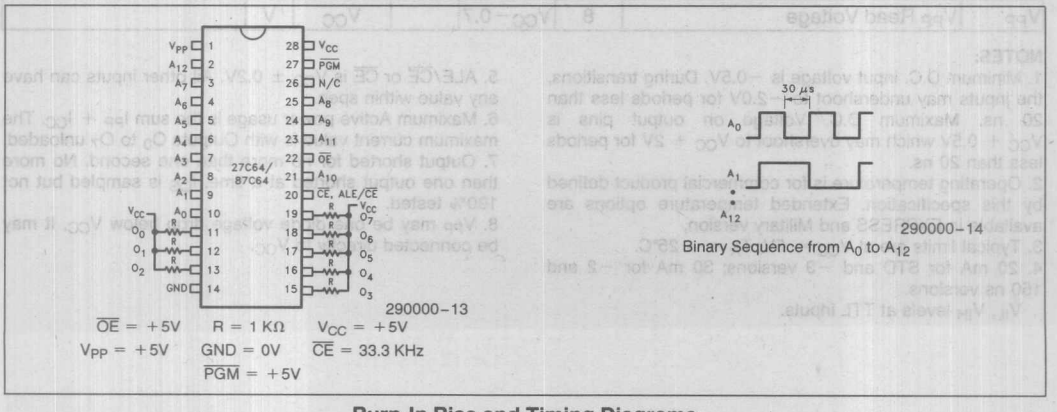
READ OPERATION

D.C. CHARACTERISTICS

Electrical Parameters of EXPRESS EPROM products are identical to standard EPROM parameters except for:

Symbol	Parameter		27C64 87C64 Min	27C64 87C64 Max	Test Conditions
I_{SB}	V_{CC} Standby Current (mA)	CMOS		0.1	$\overline{CE} = V_{CC}$, $\overline{OE} = V_{IL}$
		TTL		1.0	$\overline{CE} = V_{IH}$, $\overline{OE} = V_{IL}$
I_{CC1}(1)	V_{CC} Active Current (mA)	TTL		20, 30	$\overline{OE} = \overline{CE} = V_{IL}$
	V_{CC} Active Current at High Temperature	TTL		20, 30	$\overline{OE} = \overline{CE} = V_{IL}$ $V_{PP} = V_{CC}$, $T_{ambient} = 85°C$

NOTE:
1. See notes 4 and 6 of Read Operation D.C. Characteristics.



Burn-In Bias and Timing Diagrams

ABSOLUTE MAXIMUM RATINGS*

Operating Temperature
During Read ..... 0°C to +70°C(2)

Temperature Under Bias ..... −10°C to +80°C

Storage Temperature ..... −65°C to +150°C

Voltage on Any Pin with
Respect to Ground ..... −2.0V to 7V(1)

Voltage on Pin A_9 with
Respect to Ground ..... −2.0V to +13.5V(1)

V_{PP} Supply Voltage with Respect to Ground
During Programming ..... −2.0V to +14V(1)

V_{CC} Supply Voltage with
Respect to Ground ..... −2.0V to +7.0V(1)

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

READ OPERATION D.C. CHARACTERISTICS $0°C \le T_A \le +70°C$

Symbol	Parameter		Notes	Min	Typ(3)	Max	Unit	Test Condition
I_{LI}	Input Leakage Current				0.01	1.0	μA	$V_{IN} = 0V$ to 5.5V
I_{LO}	Output Leakage Current					±10	μA	$V_{OUT} = 0V$ to 5.5V
I_{PP_1}	V_{PP} Current Read		6			100	μA	$V_{PP} = V_{CC}$
I_{SB}	V_{CC} Current Standby with Inputs—	CMOS	5			100	μA	$\overline{CE} = V_{CC}$
		TTL	4			1.0	mA	$\overline{CE} = V_{IH}$
I_{CC_1}	V_{CC} Current Active		4, 6			20, 30	mA	$\overline{CE} = V_{IL}$, f = 5 MHz, I_{OUT} = 0 mA
V_{IL}	Input Low Voltage (±10% Supply) (TTL)			−0.5		0.8	V	$V_{PP} = V_{CC}$
	Input Low Voltage (CMOS)			−0.2		0.2		
V_{IH}	Input High Voltage (±10% Supply) (TTL)			2.0		$V_{CC}+0.5$	V	$V_{PP} = V_{CC}$
	Input High Voltage (CMOS)			$V_{CC}-0.2$		$V_{CC}+0.2$		
V_{OL}	Output Low Voltage					0.45	V	I_{OL} = 2.1 mA
V_{OH}	Output High Voltage			3.5			V	I_{OH} = −2.5 mA
I_{OS}	Output Short Circuit Current		7			100	mA	
V_{PP}	V_{PP} Read Voltage		8	$V_{CC}-0.7$		V_{CC}	V	

NOTES:

1. Minimum D.C. input voltage is −0.5V. During transitions, the inputs may undershoot to −2.0V for periods less than 20 ns. Maximum D.C. Voltage on output pins is V_{CC} + 0.5V which may overshoot to V_{CC} + 2V for periods less than 20 ns.
2. Operating temperature is for commercial product defined by this specification. Extended temperature options are available in EXPRESS and Military version.
3. Typical limits are at $V_{CC} = 5V$, $T_A = +25°C$.
4. 20 mA for STD and −3 versions; 30 mA for −2 and 150 ns versions.
   V_{IL}, V_{IH} levels at TTL inputs.
5. ALE/$\overline{CE}$ or $\overline{CE}$ is V_{CC} ± 0.2V. All other inputs can have any value within spec.
6. Maximum Active power usage is the sum I_{PP} + I_{CC}. The maximum current value is with Outputs O_0 to O_7 unloaded.
7. Output shorted for no more than one second. No more than one output shorted at a time. I_{OS} is sampled but not 100% tested.
8. V_{PP} may be one diode voltage drop below V_{CC}. It may be connected directly to V_{CC}.

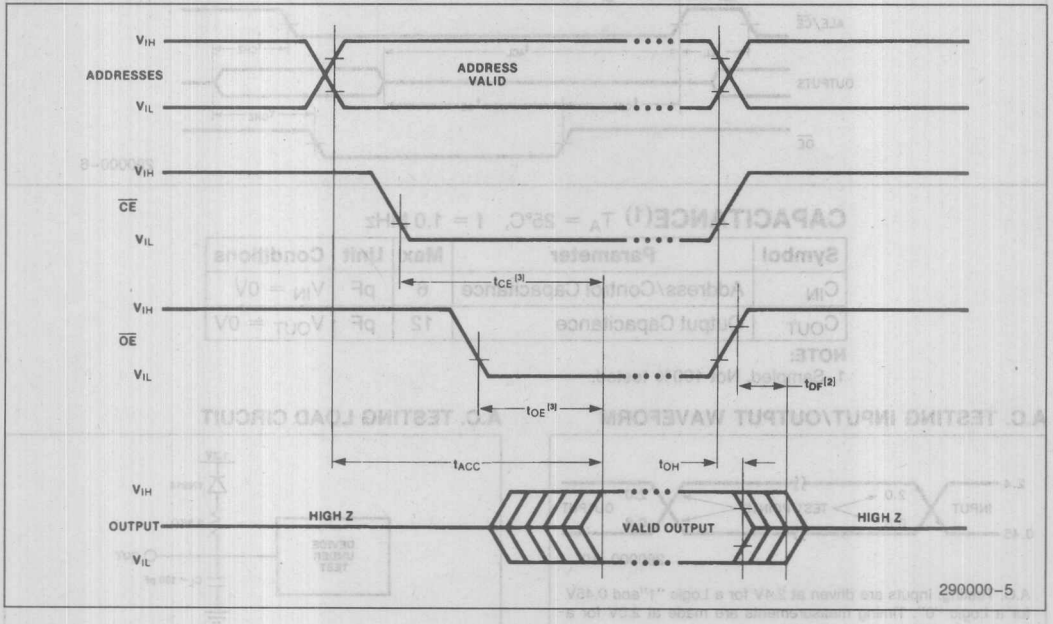
READ OPERATION

A.C. CHARACTERISTICS 27C64[1] $0°C \le T_A \le +70°C$

Versions (3)		$V_{CC} \pm 5\%$		27C64-1 N27C64-1 P27C64-1		27C64-2 N27C64-2 P27C64-2		27C64 N27C64		27C64-3 N27C64-3		Unit
		$V_{CC} \pm 10\%$		27C64-15 N27C64-15 P27C64-15		27C64-20 N27C64-20 P27C64-20		27C64-25 N27C64-25		27C64-30 N27C64-30		
Symbol	**Characteristic**			**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	
t_{ACC}	Address to Output Delay				150		200		250		300	ns
t_{CE}	$\overline{CE}$ to Output Delay				150		200		250		300	ns
t_{OE}	$\overline{OE}$ to Output Delay				75		75		100		120	ns
t_{DF}[2]	$\overline{OE}$ High to Output High Z				35		55		60		105	ns
t_{OH}[2]	Output Hold from Addresses, $\overline{CE}$ or $\overline{OE}$ Change-Whichever is First			0		0		0		0		ns

NOTES:
1. A.C. characteristics tested at $V_{IH} = 2.4V$ and $V_{IL} = 0.45V$.
   Timing measurements made at $V_{OL} = 0.8V$ and $V_{OH} = 2.0V$.
2. Guaranteed and sampled.
3. Model Number Prefixes: No prefix = Cerdip; P = Plastic DIP; N = PLCC.

A.C. WAVEFORMS 27C64

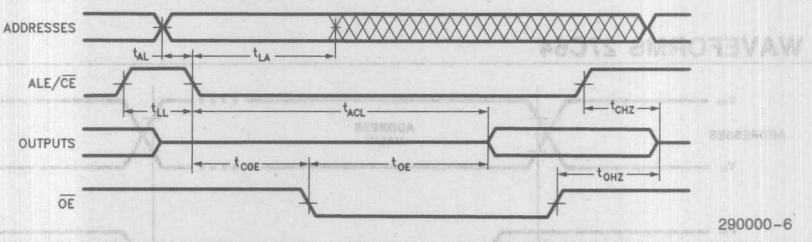


NOTES:
1. Typical values are for $T_A = 25°C$ and nominal supply voltages.
2. This parameter is only sampled and is not 100% tested.
3. $\overline{OE}$ may be delayed up to $t_{CE} - t_{OE}$ after the falling edge of $\overline{CE}$ without impact on t_{CE}.

Versions (3)		V_{CC} ±5%		87C64-1 N87C64-1 P87C64-1		87C64-2 N87C64-2 P87C64-2		87C64 N87C64		87C64-3 N87C64-3		Unit
		V_{CC} ±10%		87C64-15 N87C64-15 P87C64-15		87C64-20 N87C64-20 P87C64-20		87C64-25 N87C64-25		87C64-30 N87C64-30		
Symbol	**Parameter**		**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	
t_{LL}	Chip Deselect Width		50		50		60		75		ns
t_{AL}	Address to $\overline{CE}$-Latch Set-up		7		20		25		30		ns
t_{LA}	Address Hold from $\overline{CE}$-LATCH		30		45		50		60		ns
t_{ACL}	CE-Latch Access Time			150		200		250		300	ns
t_{OE}	Output Enable to Output Valid			75		75		100		120	ns
t_{COE}	ALE/$\overline{CE}$ to Output Enable		30		45		50		60		ns
t_{CHZ}(2)	Chip Deselect to Output in High Z			45		50		60		75	ns
t_{OHZ}(2)	Output Disable to Output in High Z			35		50		60		75	ns

NOTES:
1. A.C. characteristics tested at V_{IH} = 2.4V and V_{IL} = 0.45V.
   Timing measurements made at V_{OL} = 0.8V and V_{OH} = 2.0V.
2. Guaranteed and sampled.
3. Model Number Prefixes: No prefix = Cerdip; P = Plastic DIP; N = PLCC.

A.C. WAVEFORMS 87C64

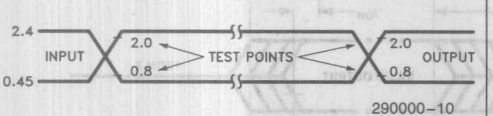


CAPACITANCE(1) T_A = 25°C, f = 1.0 MHz

Symbol	Parameter	Max	Unit	Conditions
C_{IN}	Address/Control Capacitance	6	pF	V_{IN} = 0V
C_{OUT}	Output Capacitance	12	pF	V_{OUT} = 0V

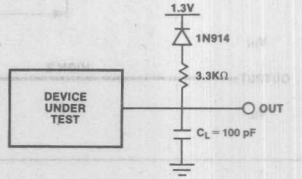
NOTE:
1. Sampled, Not 100% tested.

A.C. TESTING INPUT/OUTPUT WAVEFORM



A.C. Testing: Inputs are driven at 2.4V for a Logic "1"and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT



C_L = 100 pF
C_L Includes Jig Capacitance

DEVICE OPERATION

The modes of operation of the 27C64/87C64 are listed in Table 1. A single 5V power supply is required in the read mode. All inputs are TTL levels except for V_{PP} and 12V on A_9 for int$_e$ligent Identifier mode.

Table 1. Mode Selection for 27C64 and 87C64

Pins / Mode	ALE/$\overline{CE}$ / $\overline{CE}$	$\overline{OE}$	$\overline{PGM}$ (7)	A_9	A_0	V_{PP} (7)	V_{CC}	Outputs
Read	V_{IL}	V_{IL}	V_{IH}	X(1)	X	V_{CC}	5.0V	D_{OUT}
Output Disable	V_{IL}	V_{IH}	V_{IH}	X	X	V_{CC}	5.0V	High Z
Standby	V_{IH}	X	X	X	X	V_{CC}	5.0V	High Z
Programming	V_{IL}	V_{IH}	V_{IL}	X	X	(4)	(4)	D_{IN}
Program Verify	V_{IL}	V_{IL}	V_{IH}	X	X	(4)	(4)	D_{OUT}
Program Inhibit	V_{IH}	X	X	X	X	(4)	(4)	HIGH Z
int$_e$ligent Identifier(3) -Manufacturer	V_{IL}	V_{IL}	V_{IH}	V_H(2)	V_{IL}	V_{CC}	V_{CC}	89 H (6) 88 H (6)
int$_e$ligent Identifier(3) -27C64	V_{IL}	V_{IL}	V_{IH}	V_H(2)	V_{IH}	V_{CC}	V_{CC}	07 H
int$_e$ligent Identifier(3, 5) -87C64	V_{IL}	V_{IL}	V_{IH}	V_H(2)	V_{IH}	V_{CC}	V_{CC}	37 H

NOTES:
1. X can be V_{IL} or V_{IH}.
2. V_H = 12.0V ± 0.5V.
3. A_1–A_8, A_{10-12} = V_{IL}.
4. See Table 2 for V_{CC} and V_{PP} voltages.
5. ALE/$\overline{CE}$ has to be toggled in order to latch in the addresses and read the signature codes.
6. The Manufacturer's identifier reads 89H for Cerdip devices; 88H for Plastic DIP and PLCC devices.
7. In Read Mode tie $\overline{PGM}$ and V_{PP} to V_{CC}.

Read Mode: 27C64

The 27C64 has two control functions, both of which must be logically active in order to obtain data at the outputs. Chip Enable ($\overline{CE}$) is the power control and should be used for device selection. Output enable ($\overline{OE}$) is the output control and should be used to gate data from the output pins. Assuming that addresses are stable, the address access time (t_{ACC}) is equal to the delay from $\overline{CE}$ to output (t_{CE}). Data is available at the outputs after a delay of t_{OE} from the falling edge of $\overline{OE}$, assuming that $\overline{CE}$ has been low and addresses have been stable for at least t_{ACC}–t_{OE}.

Read Mode: 87C64

The 87C64 was designed to reduce the hardware interface requirements when incorporated in processor systems with multiplexed address-data busses. Chip count (and therefore power and board space) can be minimized when the 87C64 is designed as shown in Figure 4. The processor's multiplexed bus (AD_{0-7}) is tied to both address and data pins of the 87C64. All address inputs of the 87C64 are latched when ALE/$\overline{CE}$ is brought low, thus eliminating the need for a separate address latch.

The 87C64 internal address latch is directly enabled through the use of the ALE/$\overline{CE}$ line. As the transition occurs on the ALE/$\overline{CE}$ from the TTL high to the low state, the last address presented at the address pins is retained. Data is then enabled onto the bus from the EPROM by the $\overline{OE}$ pin.

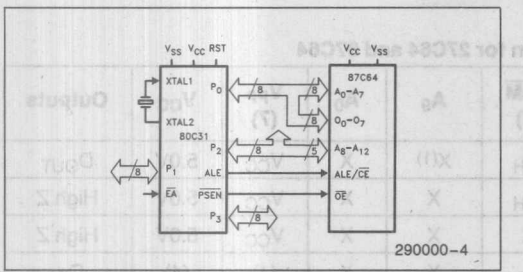


Figure 4. 80C31 with 87C64 System Configuration

Standby Mode

The 27C64 and 87C64 have Standby modes which reduce the maximum V_{CC} current to 100 μA. Both are placed in the Standby mode when $\overline{CE}$ or ALE/$\overline{CE}$ are in the CMOS-high state. When in the Standby mode, the outputs are in a high impedance state, independent of the $\overline{OE}$ input.

Two Line Output Control

Because EPROMs are usually used in larger memory arrays, Intel has provided 2 control lines which accommodate this multiple memory connection. The two control lines allow for:

a) the lowest possible memory power dissipation, and

b) complete assurance that output bus contention will not occur.

To use these two control lines most efficiently, $\overline{CE}$ (or ALE/$\overline{CE}$) should be decoded and used as the primary device selecting function, while $\overline{OE}$ should be made a common connection to all devices in the array and connected to the $\overline{READ}$ line from the system control bus. This assures that all deselected memory devices are in their low power standby mode and that the output pins are active only when data is desired from a particular memory device.

SYSTEM CONSIDERATIONS

The power switching characteristics of EPROMs require careful decoupling of the devices. The supply current, I_{CC}, has three segments that are of interest to the system designer—the standby current level, the active current level, and the transient current peaks that are produced by the falling and rising edges of Chip Enable. The magnitude of these transient and inductive current peaks is dependent on the output capacitive and inductive loading of the device. The associated transient voltage peaks can be suppressed by complying with Intel's Two-Line Control, and by properly selected decoupling capacitors. It is recommended that a 0.1 μF ceramic capacitor be used on every device between V_{CC} and GND. This should be a high frequency capacitor for low inherent inductance and should be placed as close to the device as possible. In addition, a 4.7 μF bulk electrolytic capacitor should be used between V_{CC} and GND for every eight devices. The bulk capacitor should be located near where the power supply is connected to the array. The purpose of the bulk capacitor is to overcome the voltage droop caused by the inductive effect of PC board-traces.

PROGRAMMING MODES

Caution: Exceeding 14V on V_{PP} will permanently damage the device.

Initially, and after each erasure, all bits of the EPROM are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be present in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The device is in the programming mode when V_{PP} is raised to its programming voltage (See Table 2) and $\overline{CE}$ (or ALE/$\overline{CE}$) and $\overline{PGM}$ are both at TTL low and $\overline{OE}$ = V_{IH}. The data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are TTL.

Program Inhibit

Programming of multiple EPROMS in parallel with different data is easily accomplished by using the Program Inhibit mode. A high-level $\overline{CE}$ (or ALE/$\overline{CE}$) or $\overline{PGM}$ input inhibits the other devices from being programmed.

Except for $\overline{CE}$ (or ALE/$\overline{CE}$), all like inputs (including $\overline{OE}$) of the parallel EPROMs may be common. A TTL low-level pulse applied to the $\overline{PGM}$ input with V_{PP} at its programming voltage and $\overline{CE}$ (or ALE/$\overline{CE}$) = V_{IL} will program the selected device.

Program Verify

A verify (read) should be performed on the programmed bits to determine that they have been correctly programmed. The verify is performed with $\overline{OE}$ and $\overline{CE}$ (or ALE/$\overline{CE}$) at V_{IL}, $\overline{PGM}$ at V_{IH}, and V_{CC} and V_{PP} at their programming voltages. Data should be verified a minimum of t_{OE} after the falling edge of $\overline{OE}$.

inteligent Identifier™ Mode

The inteligent Identifier Mode allows the reading out of a binary code from an EPROM that will identify its manufacturer and type. This mode is intended for use by programming equipment for the purpose of automatically matching the device to be programmed with its corresponding programming algorithm. This mode is functional in the 25°C ±5°C ambient temperature range that is required when programming the device.

To activate this mode, the programming equipment must force 11.5V to 12.5V on address line A9 of the EPROM. Two identifier bytes may then be sequenced from the device outputs by toggling address line A0 from V_{IL} to V_{IH}. All other address lines must be held at V_{IL} during the inteligent Identifier Mode.

Byte 0 (A0 = V_{IL}) represents the manufacturer code and byte 1 (A0 = V_{IH}) the device identifier code. These two identifier bytes are given in Table 1. ALE/$\overline{CE}$ of the 87C64 has to be toggled in order to latch in the addresses and read the Signature Codes.

ERASURE CHARACTERISTICS (FOR CERDIP EPROMS)

The erasure characteristics are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000Å range. Data shows that constant exposure to room level fluorescent lighting could erase the EPROM in approximately 3 years, while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the device is to be exposed to these types of lighting conditions for extended periods of time, opaque labels should be placed over the window to prevent unintentional erasure.

The recommended erasure procedure is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity × exposure time) for erasure should be a minimum of 15 Wsec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000 μW/cm^2 power rating. The EPROM should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated dose an EPROM can be exposed to without damage is 7258 Wsec/cm^2 (1 week @ 12000 μW/cm^2). Exposure of the device to high intensity UV light for longer periods may cause permanent damage.

CHMOS NOISE CHARACTERISTICS

Special EPI processing techniques have enabled Intel to build CHMOS with features adding to system reliability. These include input/output protection to latch-up. Each of the data and address pins will not latch-up with currents up to 100 mA and voltages from −1V to V_{CC} + 1V.

Additionally, the V_{PP} (programming) pin is designed to resist latch-up to the 14V maximum device limit.

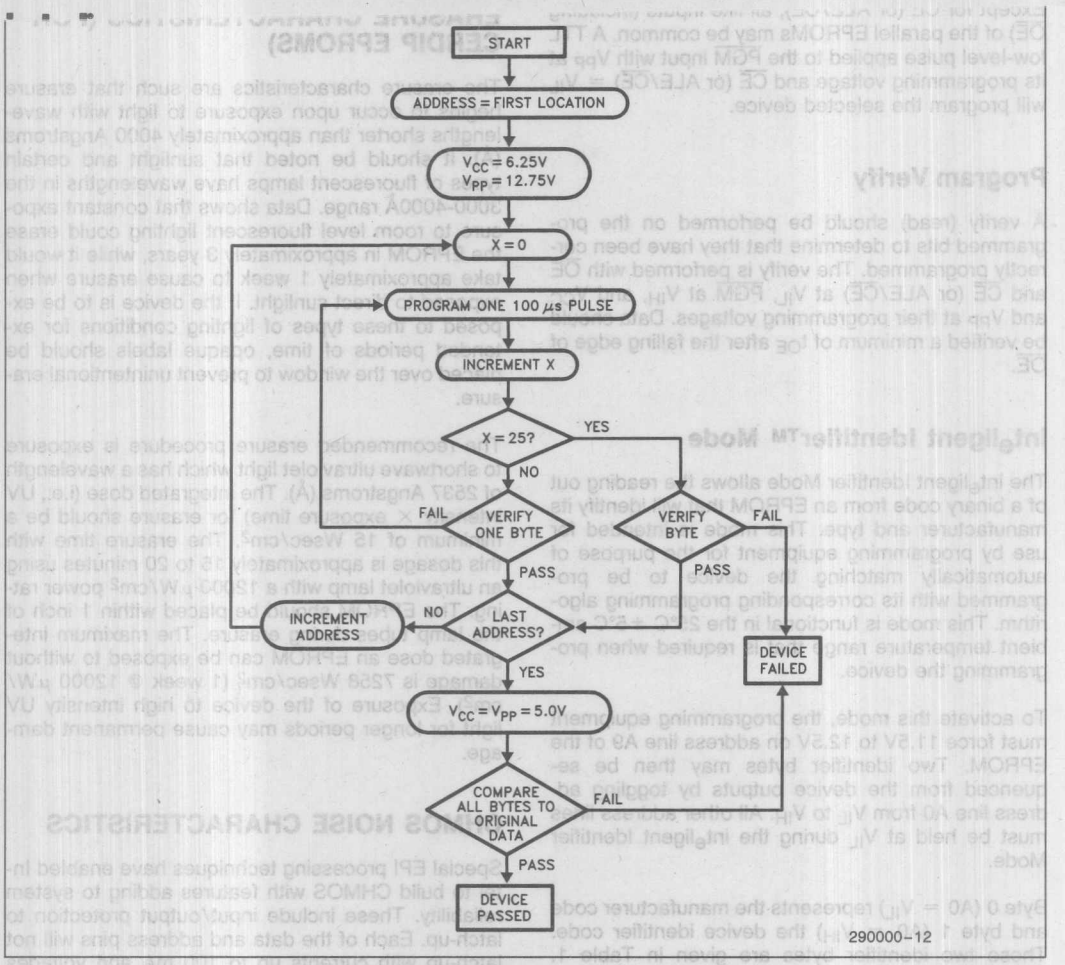


Figure 5. Quick-Pulse Programming™ Algorithm

Quick-Pulse Programming™ Algorithm

Intel's 27C64 and 87C64 EPROMs can now be programmed using the Quick-Pulse Programming Algorithm, developed by Intel to substantially reduce the throughput time in the production environment. This algorithm allows these devices to be programmed in under one second, almost a hundred fold improvement over previous algorithms. Actual programming time is a function of the PROM programmer being used.

The Quick-Pulse Programming Algorithm uses initial pulses of 100 microseconds followed by a byte verification to determine when the address byte has been successfully programmed. Up to 25 100 μs pulses per byte are provided before a failure is recognized. A flowchart of the Quick-Pulse Programming Algorithm is shown in Figure 5.

For the Quick Pulse Programming Algorithm, the entire sequence of programming pulses and byte verifications is performed at V_{CC} = 6.25V and V_{PP} at 12.75V. When programming of the EPROM has been completed, all bytes should be compared to the original data with V_{CC} = V_{PP} = 5.0V.

D.C. PROGRAMMING CHARACTERISTICS (27C64/87C64) T_A = 25°C ±5°C

Table 2

Symbol	Parameter	Limits			Test Conditions (Note 1)
		Min	Max	Unit	
I_{LI}	Input Current (All Inputs)		1.0	µA	$V_{IN} = V_{IL}$ or V_{IH}
V_{IL}	Input Low Level (All Inputs)	−0.1	0.8	V	
V_{IH}	Input High Level	2.0	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage During Verify		0.45	V	I_{OL} = 2.1 mA
V_{OH}	Output High Voltage During Verify	3.5		V	I_{OH} = −2.5 mA
I_{CC2}(3)	V_{CC} Supply Current		30	mA	
I_{PP2}(3)	V_{PP} Supply Current (Program)		30	mA	$\overline{CE} = V_{IL}$
V_{ID}	A_9 intelligent Identifier Voltage	11.5	12.5	V	
V_{PP}	Programming Voltage	12.5	13.0	V	
V_{CC}	Supply Voltage During Programming	6.0	6.5	V	

A.C. PROGRAMMING CHARACTERISTICS 27C64

T_A = 25°C ±5°C, See Table 2 for V_{CC} and V_{PP} Voltages

Symbol	Parameter	Limits				Conditions (Note 1)
		Min	Typ	Max	Unit	
t_{AS}	Address Setup Time	2			µs	
t_{OES}	$\overline{OE}$ Setup Time	2			µs	
t_{DS}	Data Setup Time	2			µs	
t_{AH}	Address Hold Time	0			µs	
t_{DH}	Data Hold Time	2			µs	
t_{DFP}	$\overline{OE}$ High to Output Float Delay	0		130	ns	(Note 2)
t_{VPS}	V_{PP} Setup Time	2			µs	
t_{VCS}	V_{CC} Setup Time	2			µs	
t_{CES}	$\overline{CE}$ Setup Time	2			µs	
t_{PW}	$\overline{PGM}$ Program Pulse Width	95	100	105	µs	Quick-Pulse
t_{OE}	Data Valid from $\overline{OE}$			150	ns	

A.C. CONDITIONS OF TEST

Input Rise and Fall Times (10% to 90%) . . . . . 20 ns

Input Pulse Levels . . . . . 0.45V to 2.4V

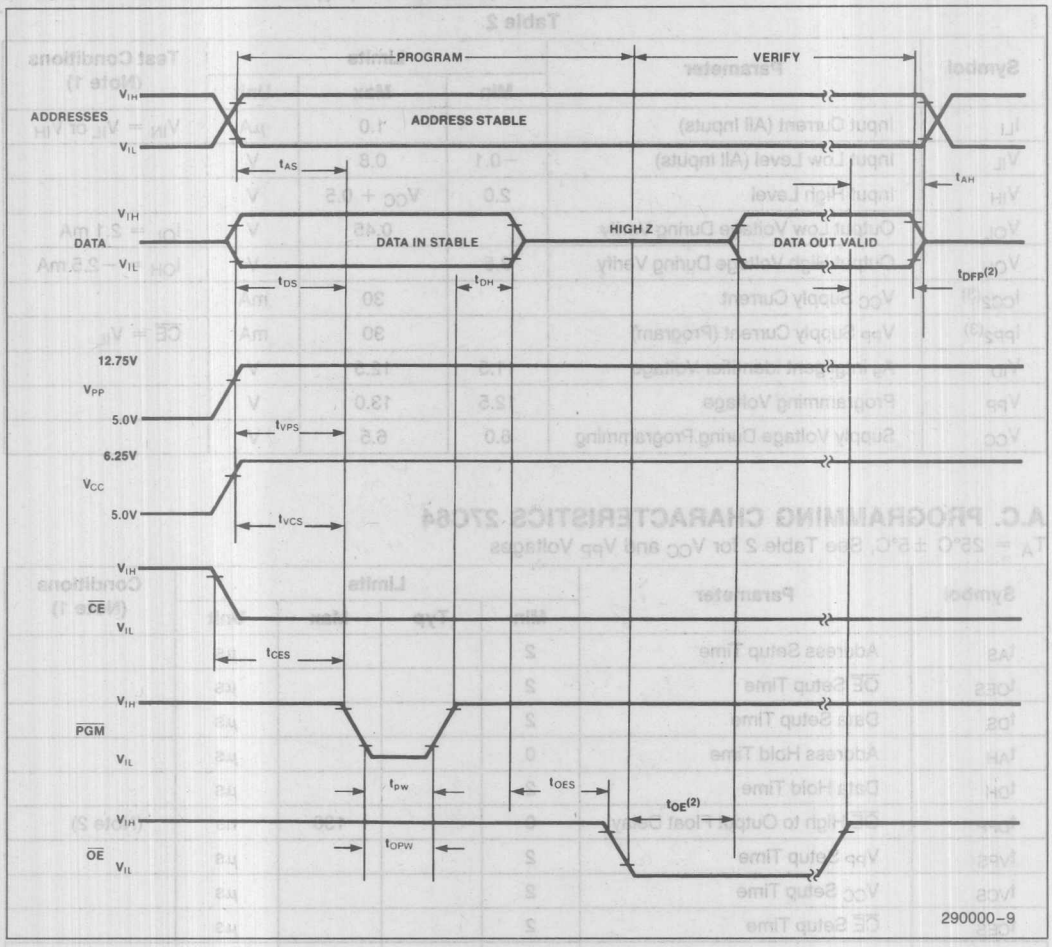
Input Timing Reference Level . . . . . 0.8V and 2.0V

Output Timing Reference Level . . . . . 0.8V and 3.5V

NOTES:

1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
2. This parameter is only sampled and is not 100% tested. Output Float is defined as the point where data is no longer driven—see timing diagram.
3. The maximum current value is with outputs O_0 to O_7 Unloaded.

PROGRAMMING WAVEFORMS 27C64



NOTES:
1. The Input Timing Reference Level is 0.8V for V_{IL} and 2V for a V_{IH}.
2. t_{OE} and t_{DFP} are characteristics of the device but must be accommodated by the programmer.
3. When programming the 27C64, a 0.1 μF capacitor is required across V_{PP} and ground to suppress spurious voltage transients which can damage the device.

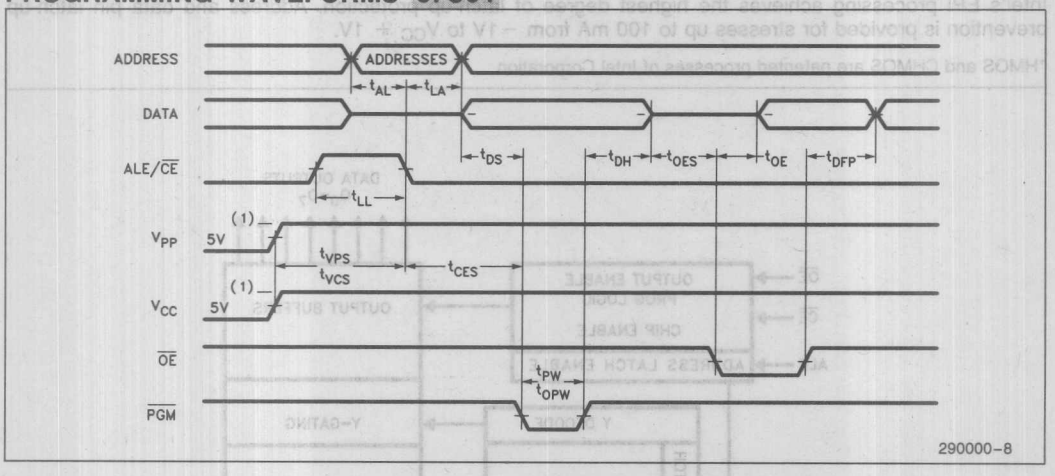
A.C. PROGRAMMING CHARACTERISTICS 87C64

T_A = 25°C ±5°C, See Table 2 for V_{CC} and V_{PP} Voltages.

Symbol	Parameter	Limits			Unit	Conditions
		Min	Typ	Max		
t_{VPS}	V_{PP} Setup Time	2			µs	
t_{VCS}	V_{CC} Setup Time	2			µs	
t_{LL}	Chip Deselect Width	2			µs	
t_{AL}	Address to Chip Select Setup	1			µs	
t_{LA}	Address Hold from Chip Select	1			µs	
t_{PW}	$\overline{PGM}$ Pulse Width	95	100	105	µs	Quick-Pulse
t_{DS}	Data Setup Time	2			µs	
t_{DFP}	$\overline{OE}$ High to Data Float	0		130	ns	
t_{OES}	Output Enable Setup Time	2			µs	
t_{OE}	Data Valid from Output Enable			150	ns	
t_{DH}	Data Hold Time	2			µs	
t_{CES}	$\overline{CE}$ Setup Time	2			µs	

NOTE:
1. Programming tolerances and test conditions are the same as 27C64.

PROGRAMMING WAVEFORMS 87C64



NOTE:
1. 12.75V V_{PP} & 6.25V V_{CC} for Quick-Pulse Programming Algorithm.

87C257
256K (32K x 8) CHMOS UV ERASABLE PROM

- **CHMOS/NMOS Microcontroller and Microprocessor Compatible**
  - **87C257-Integrated Address Latch**
  - **Universal 28 Pin Memory Site, 2-line Control**
- **Low Power Consumption**
- **High Performance Speeds**
  - **170 ns Maximum Access Time**
- **Noise Immunity Features**
  - **$\pm$10% V_{CC} Tolerance**
  - **Maximum Latch-up Immunity Through EPI Processing**
- **New Quick-Pulse Programming™ Algorithm**
  - **4 Second Programming**
- **Available in 28-Pin Cerdip Package**
  (See Packaging Spec., Order #231369)

Intel's 87C257 CHMOS EPROM is a 256K-bit 5V only memory organized as 32,768 8-bit words. It employs advanced CHMOS*II-E circuitry for systems requiring low power, high speed performance, and noise immunity. The 87C257 is optimized for compatibility with multiplexed address/data bus microcontrollers such as Intel's 16 MHz 8051- and 8096- families.

The 87C257 incorporates latches on all address inputs to minimize chip count, reduce cost, and simplify design of multiplexed bus systems. The 87C257's internal address latch allows address and data pins to be tied directly to the processor's multiplexed address/data pins. Address information (inputs A_0–A_{14}) is latched early in the memory-fetch cycle by the falling edge of the ALE input. Subsequent address information is ignored while ALE remains low. The EPROM can then pass data (from pins O_0–O_7) on the same bus during the last part of the memory-fetch cycle.

The 87C257 is offered in a ceramic DIP package, providing flexibility in prototyping and R&D environments. The 87C257 employs the Quick-Pulse Programming™ Algorithm for fast and reliable programming.

Intel's EPI processing achieves the highest degree of latch-up protection. Address and data pin latch-up prevention is provided for stresses up to 100 mA from -1V to $V_{CC} + 1$V.

*HMOS and CHMOS are patented processes of Intel Corporation.

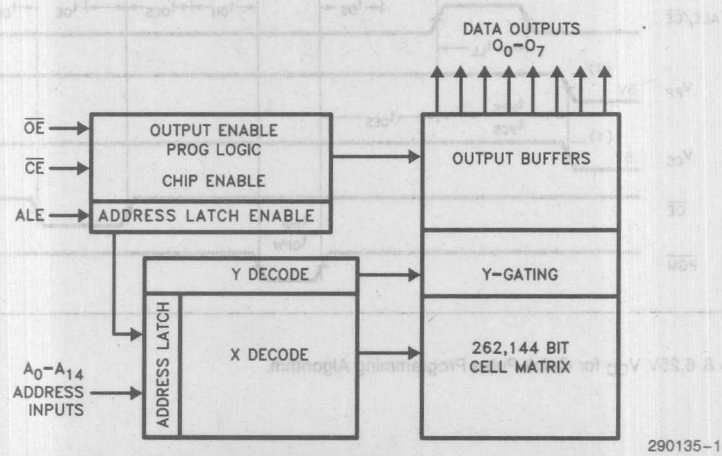


Figure 1. Block Diagram

October 1987
Order Number: 290135-002

Pin Names

A_0–A_{14}	ADDRESSES
O_0–O_7	OUTPUTS
$\overline{OE}$	OUTPUT ENABLE
$\overline{CE}$	CHIP ENABLE
ALE/V_{PP}	Address Latch Enable/V_{PP}
N.C.	NO CONNECT

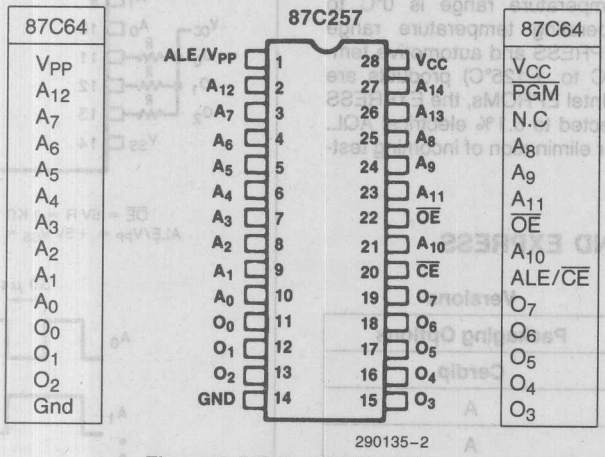


Figure 2. DIP Pin Configuration

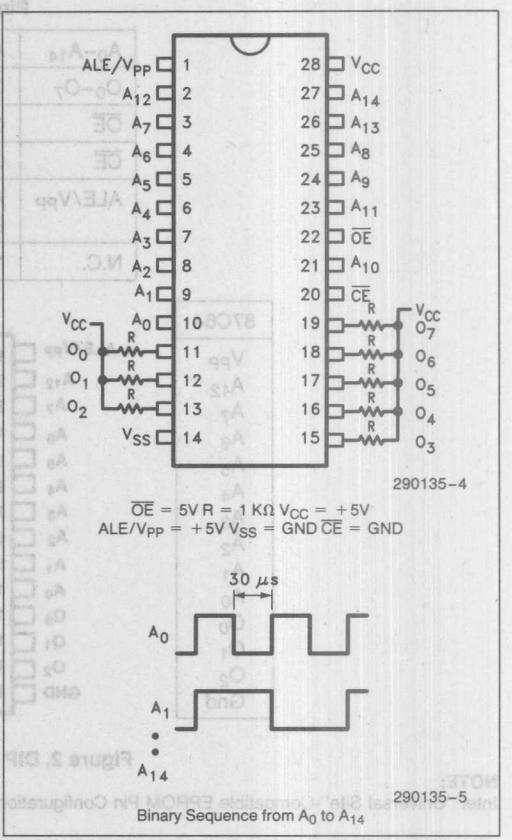
NOTE:
Intel "Universal Site"-Compatible EPROM Pin Configurations are Shown in the Blocks Adjacent.

EXTENDED TEMPERATURE (EXPRESS) EPROMs

The Intel EXPRESS EPROM family receives additional processing to enhance product characteristics. EXPRESS processing is available for several EPROM densities allowing the appropriate memory size to match system applications. EXPRESS EPROMs are available with 168 ±8 hour, 125°C dynamic burn-in using Intel's standard bias configuration. This process meets or exceeds most industry burn-in specifications. The standard EXPRESS EPROM operating temperature range is 0°C to +70°C. Extended operating temperature range (−40°C to +85°C) EXPRESS and automotive temperature range (−40°C to +125°C) products are also available. Like all Intel EPROMs, the EXPRESS EPROM family is inspected to 0.1% electrical AQL. This allows reduction or elimination of incoming testing.

AUTOMOTIVE AND EXPRESS OPTIONS

Speed Versions	Versions: Packaging Options
	Cerdip
−200V05	A
−250V10	A
−250V05	A



Burn-In Bias and Timing Diagrams

AUTOMOTIVE AND EXPRESS EPROM PRODUCT FAMILY

PRODUCT DEFINITIONS

Type	Operating Temperature (°C)	Burn-in 125°C (hr)
Q	0°C to +70°C	168 ±8
T	−40°C to +85°C	NONE
L	−40°C to +85°C	168 ±8
A	−40°C to +125°C	NONE
B	−40°C to +125°C	168 ±8

ABSOLUTE MAXIMUM RATINGS*

Operating Temperature During Read ..... 0°C to + 70°C(2)
Temperature Under Bias ..... −10°C to +80°C(2)
Storage Temperature ..... −65°C to +150°C
Voltage on any Pin with Respect to Ground ..... −2V to +7V(1)
Voltage on A_9 with Respect to Ground ..... −2V to +13.5V(1)
V_{PP} Supply Voltage with Respect to Ground During Programming ..... −2V to +14.0V(1)
V_{CC} Supply Voltage with Respect to Ground ..... −2V to +7.0V(1)

Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

READ OPERATION

D.C. CHARACTERISTICS TTL and NMOS Inputs

Symbol	Parameter		Notes	Min	Typ(3)	Max	Units	Test Condition
I_{LI}	Input Load Current				0.01	1.0	μA	V_{IN} = 0V to 5.5V
I_{LO}	Output Leakage Current					±10	μA	V_{OUT} = 0V to 5.5V
I_{SB}	V_{CC} Current Standby with Inputs—	Switching				10	mA	$\overline{CE}$ = ALE = V_{IH}
		Stable				1.0	mA	$\overline{CE}$ = V_{IH}, ALE = V_{IL}
I_{CC_1}	V_{CC} Current Active		5			30	mA	$\overline{CE}$ = V_{IL}, ALE = V_{IH} f = 5 MHz, I_{OUT} = 0 mA
V_{IL}	Input Low Voltage (±10% Supply)		1	−0.5		0.8	V	
V_{IH}	Input High Voltage (±10% Supply)			2.0		V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage					0.45	V	I_{OL} = 2.1 mA
V_{OH}	Output High Voltage			2.4			V	I_{OH} = −400 μA
I_{OS}	Output Short Circuit Current		6			100	mA	

D.C. CHARACTERISTICS CMOS Inputs

Symbol	Parameter		Notes	Min	Typ(3)	Max	Units	Test Condition
I_{LI}	Input Load Current				0.01	1.0	μA	V_{IN} = 0V to 5.5V
I_{LO}	Output Leakage Current					±10	μA	V_{OUT} = 0V to 5.5V
I_{SB}	V_{CC} Current Standby with Inputs—	Switching	4			6	mA	$\overline{CE}$ = ALE = V_{CC}
		Stable				100	μA	$\overline{CE}$ = V_{CC}, ALE = GND
I_{CC_1}	V_{CC} Current Active		5			15	mA	$\overline{CE}$ = V_{IL}, ALE = V_{IH} f = 5 MHz, I_{OUT} = 0 mA
V_{IL}	Input Low Voltage (±10% Supply)			−0.2		0.8	V	
V_{IH}	Input High Voltage (±10% Supply)			0.7 V_{CC}		V_{CC} + 0.2	V	
V_{OL}	Output Low Voltage					0.4	V	I_{OL} = 2.1 mA
V_{OH}	Output High Voltage			V_{CC} − 0.8			V	I_{OH} = −2.5 mA
I_{OS}	Output Short Circuit Current		6			100	mA	

NOTES:
1. Minimum D.C. input voltage is −0.5V. During transitions, the inputs may undershoot to −2.0V for periods less than 20 ns. Maximum D.C. voltage on output pins is V_{CC} + 0.5V which may overshoot to V_{CC} + 2V for periods less than 20 ns.
2. Operating temperature is for commercial product defined by this specification. Extended temperature options are available in EXPRESS and Automotive versions.
3. Typical limits are at V_{CC} = 5V, T_A = +25°C.
4. $\overline{CE}$ is V_{CC} ±0.2V. All other inputs can have any value within spec.
5. Maximum current value is with outputs O_0 to O_7 unloaded.
6. Output shorted for no more than one second. No more than one output shorted at a time. I_{OS} is sampled but not 100% tested.

READ OPERATION

A.C. CHARACTERISTICS(1) $0°C \le T_A \le +70°C$

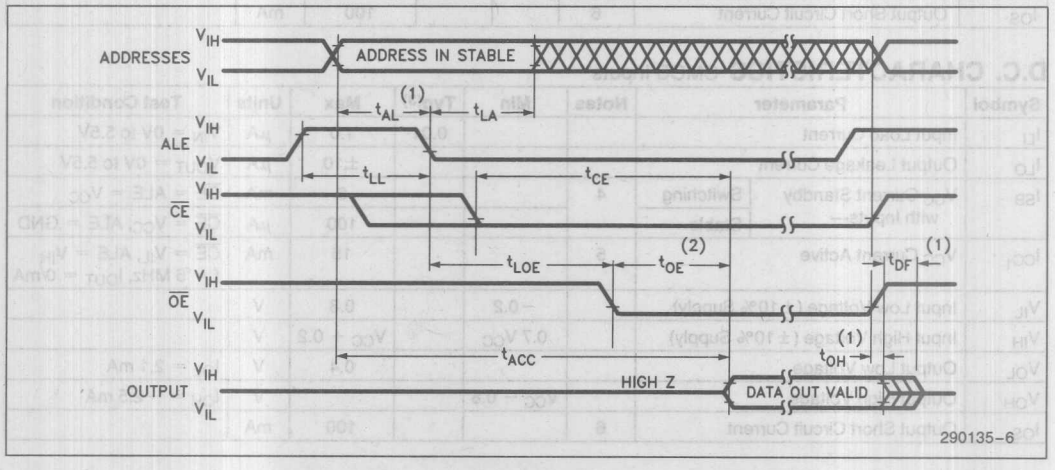
Versions(3)		$V_{CC} \pm 5\%$	87C257-170V05		87C257-200V05		87C257-250V05		
		$V_{CC} \pm 10\%$			87C257-200V10		87C257-250V10		Units
Symbol	Characteristic		Min	Max	Min	Max	Min	Max	
t_{ACC}	Address to Output Delay			170		200		250	ns
t_{CE}	$\overline{CE}$ to Output Delay			170		200		250	ns
t_{OE}	$\overline{OE}$ to Output Delay			70		75		100	ns
t_{DF}(2)	$\overline{OE}$ High to Output High Z			35		40		55	ns
t_{OH}(2)	Output Hold from Addresses, $\overline{CE}$ or $\overline{OE}$ Change-Whichever is First		0		0		0		ns
t_{LL}	Latch Deselect Width		35		55		60		ns
t_{AL}(2)	Address to Latch Set-Up		7		15		25		ns
t_{LA}	Address Hold from LATCH		20		30		40		ns
t_{LOE}	ALE to Output Enable		20		30		40		ns

NOTES:
1. See A.C. Testing Input/Output Waveforms for timing measurements.
2. Guaranteed and sampled.
3. Model Number Prefixes: No Prefix = CERDIP.

A.C. CONDITIONS OF TEST

Input Rise and Fall Times (10% to 90%) ..... 10 ns
Input Pulse Levels ..... V_{OL} to V_{OH}
Input Timing Reference Level 1.5V
Output Timing Reference Level ..... V_{IL} and V_{IH}

A.C. WAVEFORMS



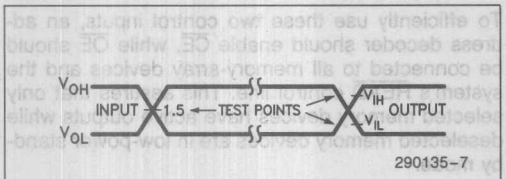
NOTES:
1. This parameter is only sampled and is not 100% tested.
2. $\overline{OE}$ may be delayed up to $t_{CE} - t_{OE}$ after the falling edge of $\overline{CE}$ without impact on t_{CE}.

CAPACITANCE[1] T_A = 25°C, f = 1.0 MHz

Symbol	Parameter	Max	Units	Conditions
C_{IN}	Address/Control Capacitance	6	pF	V_{IN} = 0V
C_{OUT}	Output Capacitance	12	pF	V_{OUT} = 0V

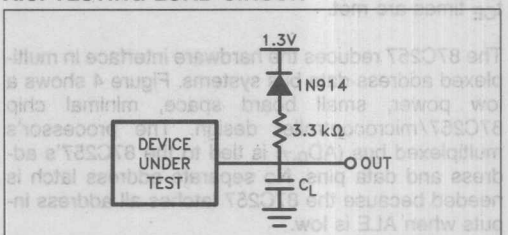
NOTE:
1. Sampled. Not 100% tested.

A.C. TESTING INPUT/OUTPUT WAVEFORM



A.C. testing inputs are driven at V_{OH} for a Logic "1" and V_{OL} for a Logic "0". Timing measurements are made at V_{IH} for a Logic "1" and V_{IL} for a Logic "0".

A.C. TESTING LOAD CIRCUIT



C_L = 100 pF
C_L Includes Jig Capacitance

DEVICE OPERATION

Table 1 lists 87C257 operating modes. Read mode requires a single 5V power supply. All input levels are TTL or CMOS except A9 in int$_e$ligent Identifier mode and V_{PP}.

Table 1. Mode Selection

Pins / Mode	$\overline{CE}$	$\overline{OE}$	A_9	A_0	ALE/V_{PP}	V_{CC}	Outputs
Read	V_{IL}	V_{IL}	X[1]	X	X	5.0V	D_{OUT}
Output Disable	V_{IL}	V_{IH}	X	X	X	5.0V	High Z
Standby	V_{IH}	X	X	X	X	5.0V	High Z
Programming	V_{IL}	V_{IH}	X	X	(Note 4)	(Note 4)	D_{IN}
Program Verify	V_{IH}	V_{IL}	X	X	(Note 4)	(Note 4)	D_{OUT}
Optional Program Verify	V_{IL}	V_{IL}	X	X	V_{CC} (Note 4)	(Note 4)	D_{OUT}
Program Inhibit	V_{IH}	V_{IH}	X	X	(Note 4)	(Note 4)	High Z
int$_e$ligent Identifier[3] -Manufacturer	V_{IL}	V_{IL}	V_H[2]	V_{IL}	X	V_{CC}	89 H
int$_e$ligent Identifier[3] -87C257	V_{IL}	V_{IL}	V_H[2]	V_{IH}	X	V_{CC}	24 H

NOTES:
1. X can be V_{IL} or V_{IH}.
2. V_H = 12.0V ±0.5V.
3. A_1–A_8, A_{10-12} = V_{IL}, A_{13-14} = X.
4. See Table 2 for V_{CC} and V_{PP} programming voltages.

Read Mode

The 87C257 has two control functions; both must be logically active to obtain data at the outputs. Chip Enable ($\overline{CE}$) is the power control and the device-select. Output enable ($\overline{OE}$) gates data to the output pins by controlling the output buffer. When the address is stable (ALE = V_{IH}) or latched (ALE = V_{IL}), the address access time (t_{ACC}) equals the delay from $\overline{CE}$ to output (t_{CE}). Outputs display valid data t_{OE} after the falling edge of $\overline{OE}$, assuming t_{ACC} and t_{CE} times are met.

The 87C257 reduces the hardware interface in multiplexed address-data bus systems. Figure 4 shows a low power, small board space, minimal chip 87C257/microcontroller design. The processor's multiplexed bus (AD_{0-7}) is tied to the 87C257's address and data pins. No separate address latch is needed because the 87C257 latches all address inputs when ALE is low.

The ALE input controls the 87C257's internal address latch. As ALE transitions from V_{IH} to V_{IL}, the last address present at the address pins is retained. The $\overline{OE}$ control can then enable EPROM data onto the bus.

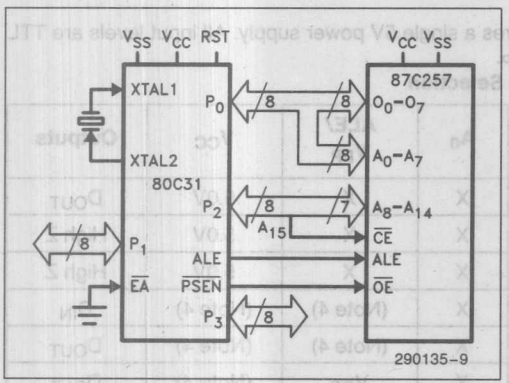


Figure 4. 80C31 with 87C257 System Configuration

Standby Mode

The standby mode substantially reduces V_{CC} current. When $\overline{CE}$ = V_{IH}, the standby mode places the outputs in a high impedance state, independent of the $\overline{OE}$ input.

Two Line Output Control

EPROMs are often used in larger memory arrays. Intel provides two contol inputs to accommodate multiple memory connections. Two-line control provides for:

a) the lowest possible memory power dissipation, and

b) complete assurance that output bus contention will not occur.

To efficiently use these two control inputs, an address decoder should enable $\overline{CE}$, while $\overline{OE}$ should be connected to all memory-array devices and the system's $\overline{READ}$ control line. This assures that only selected memory devices have active outputs while deselected memory devices are in low-power standby mode.

SYSTEM CONSIDERATIONS

EPROM power switching characteristics require careful device decoupling. System designers are interested in three supply current (ICC) issues—standby current levels, active current levels, and transient current peaks produced by falling and rising edges of Chip Enable. Transient current magnitudes depend on the device outputs' capacitive and inductive loading. Two-Line Control and proper decoupling capacitor selection will suppress transient voltage peaks. Each device should have a 0.1 μF ceramic capacitor connected between its V_{CC} and GND. This high frequency, low inherent-inductance capacitor should be placed as close as possible to the device. Additionally, for every eight devices, a 4.7 μF electrolytic capacitor should be placed between V_{CC} and GND at the array's power supply connection. The bulk capacitor will overcome voltage slumps caused by PC board trace inductances.

PROGRAMMING MODES

Caution: Exceeding 14V on V_{PP} will permanently damage the device.

Initially, and after each erasure, all EPROM bits are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" are programmed, the data word

can contain both "1s" and "0s". Ultraviolet light erasure is the only way to change "0s" to "1s".

The programming mode is entered when V_{PP} is raised to its programming voltage (see Table 2). Data is programmed by applying an 8-bit word to the output pins (O_{0-7}). Pulsing $\overline{CE}$ to TTL-low while $\overline{OE} = V_{IH}$ will program data. TTL levels are required for address and data inputs.

Program Inhibit

The Program Inhibit mode allows parallel programming of multiple EPROMs with different data. With V_{PP} at its programming voltage, a $\overline{CE}$-low pulse programs the desired EPROM. $\overline{CE}$-high inputs inhibit programming of non-targeted devices. Except for $\overline{CE}$ and $\overline{OE}$, parallel EPROMs may have common inputs.

Program Verify

With V_{PP} and V_{CC} at their programming voltages, a verify (read) determines that bits are correctly programmed. The verify is performed with $\overline{CE} = V_{IH}$ and $\overline{OE} = V_{IL}$. Valid data is available t_{OE} after $\overline{OE}$ falls low.

Optional Program Verify

The optional verify allows parallel programming and verification when several devices share a common bus. It is performed with $\overline{CE} = \overline{OE} = V_{IL}$ and $V_{PP} = V_{CC} = 6.25V$. The normal read mode is then used for program verify. Outputs will tri-state depending on $\overline{OE}$ and $\overline{CE}$.

inteligent Identifier™ Mode

The inteligent Identifier Mode will determine an EPROM's manufacturer and device type. Programming equipment can automatically match a device with its proper programming algorithm.

This mode is activated when programming equipment forces 12V ±0.5V on the EPROM's A_9 address line. With A_1-A_8, $A_{10}-A_{12} = V_{IL}$ (A_{13-14} are don't care), address line $A_0 = V_{IL}$ will present the manufacturer's code and $A_0 = V_{IH}$ the device code (see Table 1). When $A_9 = V_H$, ALE need not be toggled to latch each identifier address. This mode functions in the 25°C ±5°C ambient temperature range required during programming.

ERASURE CHARACTERISTICS (FOR CERDIP EPROMS)

Exposure to light of wavelength shorter than 4000 Angstroms (Å) begins EPROM erasure. Sunlight and some fluorescent lamps have wavelengths in the 3000–4000Å range. Constant exposure to room-level fluorescent light can erase an EPROM in about 3 years (about 1 week for direct sunlight). Opaque labels over the window will prevent unintentional erasure under these lighting conditions.

The recommended erasure procedure is exposure to 2537Å ultraviolet light. The minimum integrated dose (intensity x exposure time) is 15 $Wsec/cm^2$. Erasure time using a 12000 $\mu W/cm^2$ ultraviolet lamp is approximately 15 to 20 minutes. The EPROM should be placed about 1 inch from the lamp. The maximum integrated dose is 7258 $Wsec/cm^2$ (1 week @ 12000 $\mu W/cm^2$). High intensity UV light exposure for longer periods can cause permanent damage.

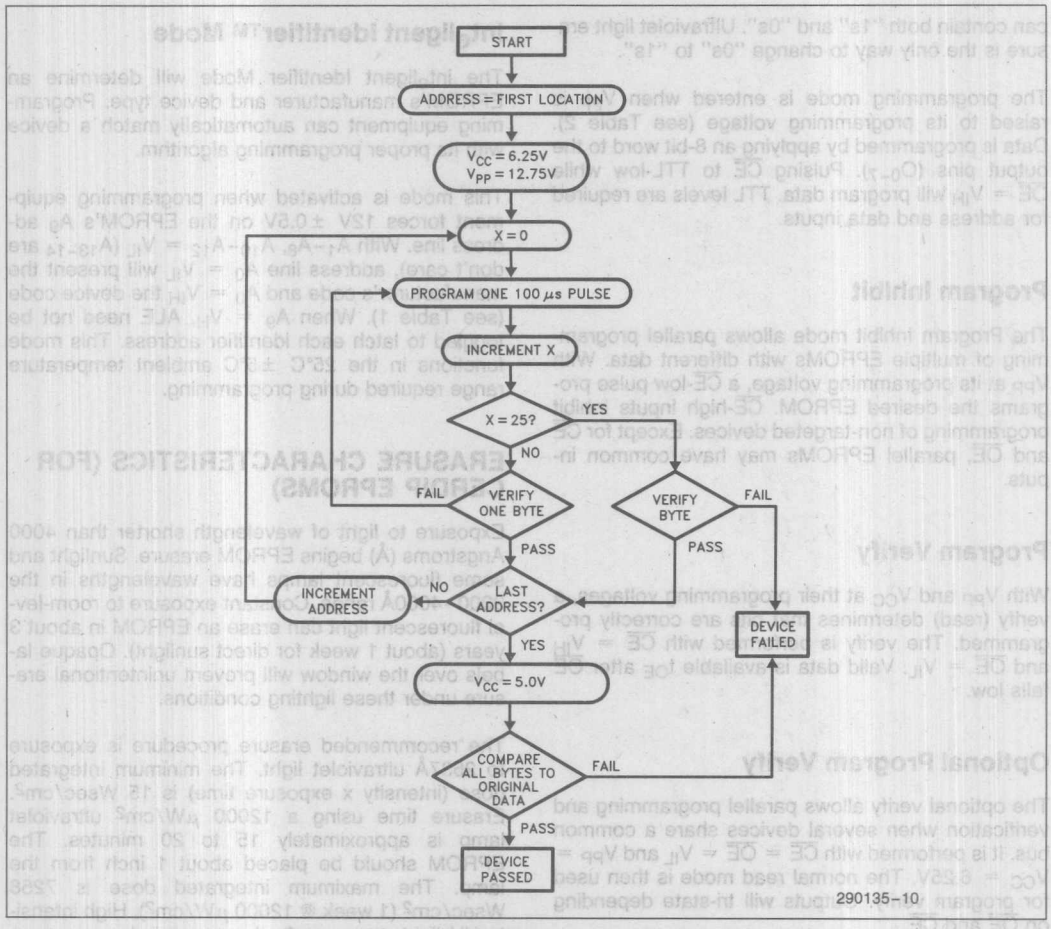


Figure 5. Quick-Pulse Programming™ Algorithm

CHMOS NOISE CHARACTERISTICS

System reliability is enhanced by Intel's CHMOS EPI-process techniques. Protection on each data and address pin prevents latch-up; even with 100 mA currents and voltages from $-1V$ to $V_{CC} + 1V$. Additionally, the V_{PP} pin is designed to resist latch-up to the 14V maximum device limit.

Quick-Pulse Programming™ Algorithm

The Quick-Pulse Programming algorithm programs Intel's 87C257 EPROM. Developed to substantially reduce production programming throughput time, this algorithm can program a 87C257 in under four seconds. Actual programming time depends on the PROM programmer used.

The Quick-Pulse Programming algorithm uses a 100 microsecond initial-pulse followed by a byte verification to determine when the addressed byte is correctly programmed. The algorithm terminates if 25 100µs pulses fail to program a byte. Figure 5 shows the Quick-Pulse Programming algorithm flowchart.

The entire program-pulse/byte-verify sequence is performed with $V_{CC} = 6.25V$ and $V_{PP} = 12.75V$. When programming is complete, all bytes should be compared to the original data with $V_{CC} = 5.0V$.

Alternate Programming

Intel's 27C256 and 27256 Quick-Pulse Programming algorithms will also program the 87C257. By overriding a check for the int$_e$ligent Identifier, older or non-upgraded PROM programmers can program the 87C257. See Intel's 27C256 and 27256 data sheets for programming waveforms of these alternate algorithms.

D.C. PROGRAMMING CHARACTERISTICS $T_A = 25°C \pm 5°C$

Table 2

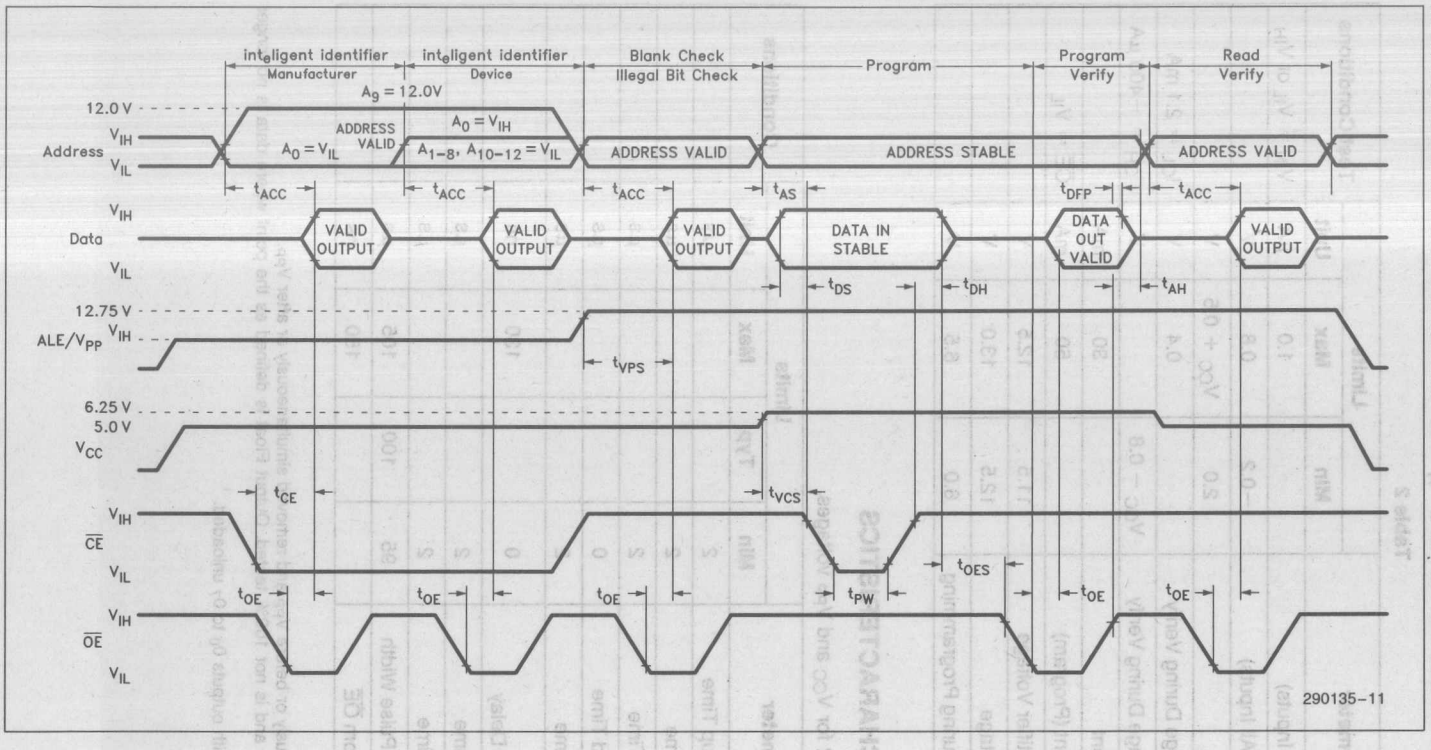
Symbol	Parameter	Limits			Test Conditions
		Min	Max	Unit	
I_{LI}	Input Current (All Inputs)		1.0	µA	$V_{IN} = V_{IL}$ or V_{IH}
V_{IL}	Input Low Level (All Inputs)	−0.2	0.8	V	
V_{IH}	Input High Level	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage During Verify		0.4	V	$I_{OL} = 2.1$ mA
V_{OH}	Output High Voltage During Verify	$V_{CC} - 0.8$		V	$I_{OH} = -400$ µA
I_{CC2}(3)	V_{CC} Supply Current		30	mA	
I_{PP2}(3)	V_{PP} Supply Current (Program)		50	mA	$\overline{CE} = V_{IL}$
V_{ID}	A_9 int$_e$ligent Identifier Voltage	11.5	12.5	V	
V_{PP}(1)	Programming Voltage	12.5	13.0	V	
V_{CC}(1)	Supply Voltage During Programming	6.0	6.5	V	

A.C. PROGRAMMING CHARACTERISTICS

$T_A = 25°C \pm 5°C$; see Table 2 for V_{CC} and V_{PP} voltages.

Symbol	Parameter	Limits				Conditions
		Min	Typ	Max	Unit	
t_{AS}	Address Setup Time	2			µs	
t_{OES}	$\overline{OE}$ Setup Time	2			µs	
t_{DS}	Data Setup Time	2			µs	
t_{AH}	Address Hold Time	0			µs	
t_{DH}	Data Hold Time	2			µs	
t_{DFP}(2)	$\overline{OE}$ High to Output Float Delay	0		130	ns	
t_{VPS}(1)	V_{PP} Setup Time	2			µs	
t_{VCS}(1)	V_{CC} Setup Time	2			µs	
t_{PW}	$\overline{CE}$ Program Pulse Width	95	100	105	µs	
t_{OE}	Data Valid from $\overline{OE}$			150	ns	

NOTES:
1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
2. This parameter is only sampled and is not 100% tested. Output Float is defined as the point where data is no longer driven—see timing diagram.
3. The maximum current value is with outputs O_0 to O_7 unloaded.



NOTES:

1. The input timing reference level is V_{IL} = 0.8V and V_{IH} = 2V.
2. t_{OE} and t_{DFP} are device characteristics but must be accommodated by the programmer.
3. To prevent device damage during programming, a 0.1 μF capacitor is required between V_{PP} and ground to suppress spurious voltage transients.
4. During programming, the address latch function is bypassed whenever V_{PP} = 12.75V or $A_9 = V_H$. When V_{PP} and A_9 are at TTL levels, the address latch function is enabled, and the device functions in read mode.
5. V_{PP} can be 12.75V during Blank Check and Final Verify; if so, $\overline{CE}$ must be V_{IH}.

UPI-452
CHMOS PROGRAMMABLE I/O PROCESSOR

83C452 - 8K × 8 Mask Programmable Internal ROM

87C452P - 8K × 8 Piggyback EPROM

80C452 - External ROM/EPROM

- 83C452/87C452P/80C452:3.5 to 16 MHz Clock Rate
- Software Compatible with the MCS-51 Family
- 128-Byte Bi-Directional FIFO Slave Interface
- Two DMA Channels
- 256 × 8-Bit Internal RAM
- 34 Additional Special Function Registers
- 40 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Boolean Processor
- Bit Addressable RAM
- 8 Interrupt Sources
- Programmable Full Duplex Serial Channel
- 64K Program Memory Space
- 64K Data Memory Space
- 68-Pin PGA
  (See Packaging Spec., Order: #231369)

The Intel UPI-452 (Universal Peripheral Interface) is a 68 pin CHMOS Slave I/O Processor with a sophisticated bi-directional FIFO buffer interface on the slave bus and a two channel DMA processor on-chip. The UPI-452 is the newest member of Intel's UPI family of products. It is a general-purpose slave I/O Processor that allows the designer to grow a customized interface solution.

The UPI-452 contains a complete 80C51 with twice the on-chip data and program memory. The sophisticated slave FIFO module acts as a buffer between the UPI-452 internal CPU and the external host CPU. To both the external host and the internal CPU, the FIFO module looks like a bi-directional bottomless buffer that can both read and write data. The FIFO manages the transfer of data independent of the UPI-452 core CPU and generates an interrupt or DMA request to either CPU, host or internal, as a FIFO service request.

The FIFO consists of two channels:the Input FIFO and the Output FIFO. The division of the FIFO module array, 128 bytes, between Input channel and Output channel is programmable by the user. Each FIFO byte has an additional logical ninth bit to distinguish between a data byte and a Data Stream Command byte. Additionally, Immediate Commands allow direct, interrupt driven, bi-directional communication between the UPI-452 internal CPU and external host CPU, bypassing the FIFO.

The on-chip DMA processor allows high speed data transfers from one writeable memory space to another. As many as 64K bytes can be transferred in a single DMA operation. Three distinct memory spaces may be used in DMA operations; Internal Data Memory, External Data Memory, and the Special Function Registers (including the FIFO IN, FIFO OUT, and Serial Channel Special Functions Registers).

September 1987
Order Number: 231428-003

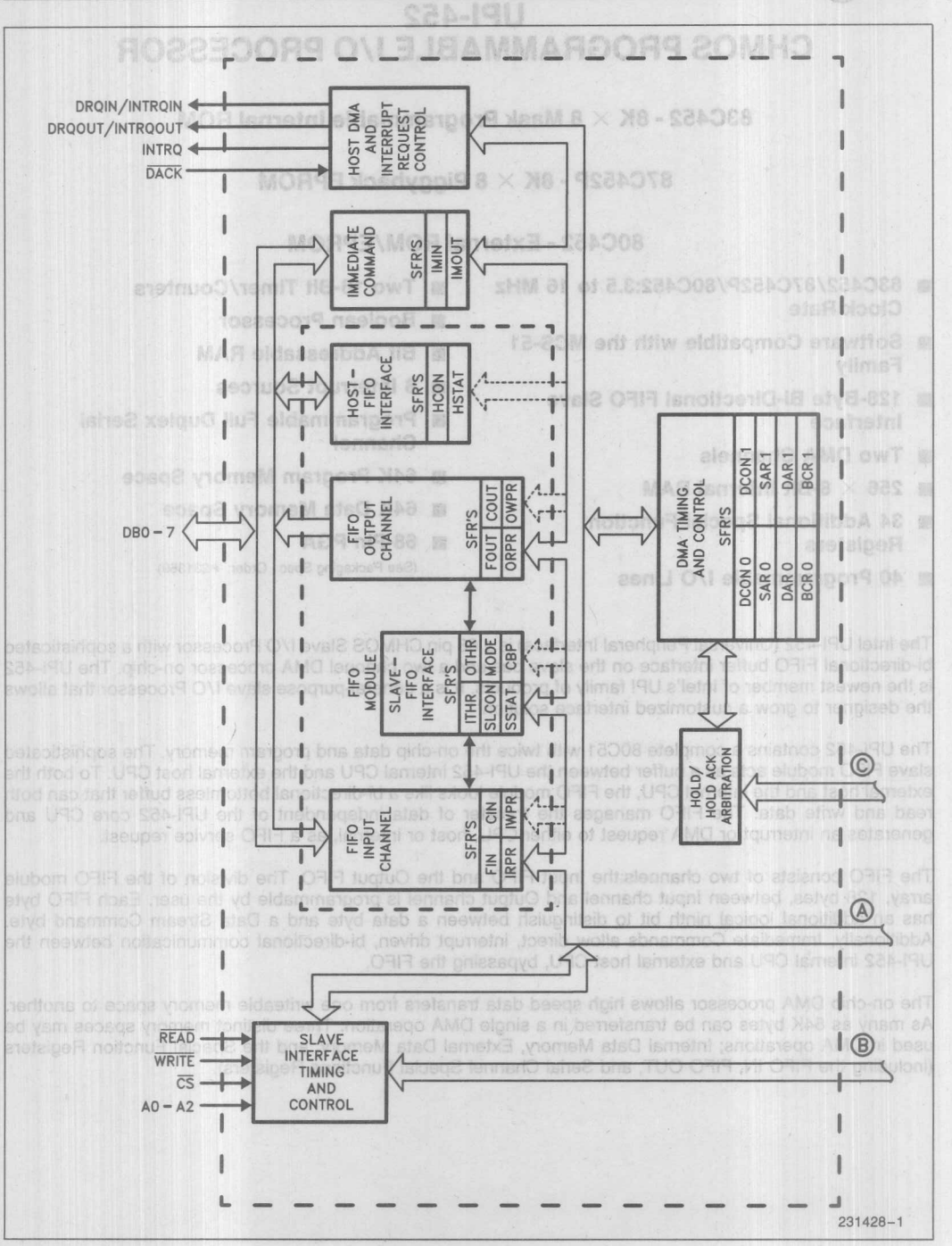


Figure 1. Architectural Block Diagram

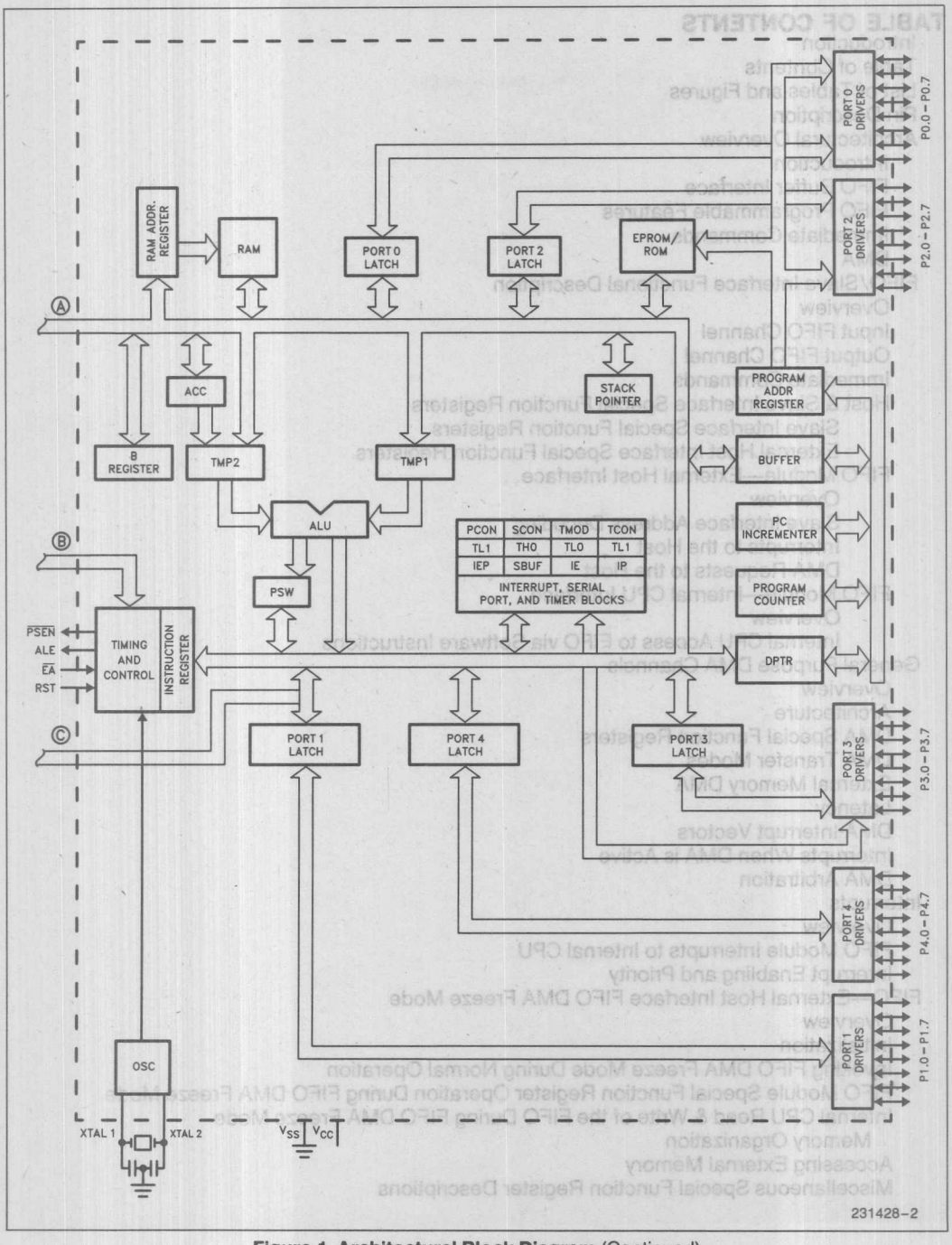


Figure 1. Architectural Block Diagram (Continued)

TABLE OF CONTENTS

- Introduction
- Table of Contents
- List of Tables and Figures
- Pin Description
- Architectural Overview
  - Introduction
  - FIFO Buffer Interface
  - FIFO Programmable Features
  - Immediate Commands
  - DMA
- FIFO/Slave Interface Functional Description
  - Overview
  - Input FIFO Channel
  - Output FIFO Channel
  - Immediate Commands
  - Host & Slave Interface Special Function Registers
    - Slave Interface Special Function Registers
    - External Host Interface Special Function Registers
  - FIFO Module—External Host Interface
    - Overview
    - Slave Interface Address Decoding
    - Interrupts to the Host
    - DMA Requests to the Host
  - FIFO Module—Internal CPU Interface
    - Overview
    - Internal CPU Access to FIFO via Software Instructions
- General Purpose DMA Channels
  - Overview
  - Architecture
  - DMA Special Function Registers
  - DMA Transfer Modes
  - External Memory DMA
  - Latency
  - DMA Interrupt Vectors
  - Interrupts When DMA is Active
  - DMA Arbitration
- Interrupts
  - Overview
  - FIFO Module Interrupts to Internal CPU
  - Interrupt Enabling and Priority
- FIFO—External Host Interface FIFO DMA Freeze Mode
  - Overview
  - Initialization
  - Invoking FIFO DMA Freeze Mode During Normal Operation
  - FIFO Module Special Function Register Operation During FIFO DMA Freeze Mode
  - Internal CPU Read & Write of the FIFO During FIFO DMA Freeze Mode
    - Memory Organization
  - Accessing External Memory
  - Miscellaneous Special Function Register Descriptions

LIST OF TABLES AND FIGURES

Figures:

1. Architectural Block Diagram
2. UPI-452 68-Pin PGA Pinout Diagram
3. UPI-452 Conceptual Block Diagram
4. UPI-452 Functional Block Diagram
5. Input FIFO Channel Functional Block Diagram
6. Output FIFO Channel Functional Block Diagram

7a. Handshake Mechanisms for Handling Immediate Command IN Flowchart

7b. Handshake Mechanisms for Handling Immediate Command OUT Flowchart

8. DMA Transfer from: External to External Memory
9. DMA Transfer from: External to Internal Memory
10. DMA Transfer from: Internal to External Memory
11. DMA Transfer Waveform: Internal to Internal Memory
12. Disabling FIFO to Host Slave Interface Timing Diagram

Tables:

1. Input FIFO Channel Registers
2. Output FIFO Channel Registers
3. UPI-452 Address Decoding
4. DMA Accessible Special Function Registers
5. DMA Mode Control - PCON SFR
6. Interrupt Priority
7. Interrupt Vector Addresses
8. Slave Bus Interface Status During FIFO DMA Freeze Mode
9. FIFO SFR's Characteristics During FIFO DMA Freeze Mode
10. Threshold SFRs Range of Values and Number of Bytes to be Transferred

11a. 80C51 Special Function Registers

11b. UPI-452 Additional Special Function Registers

12. Program Status Word (PSW)
13. PCON Special Function Register

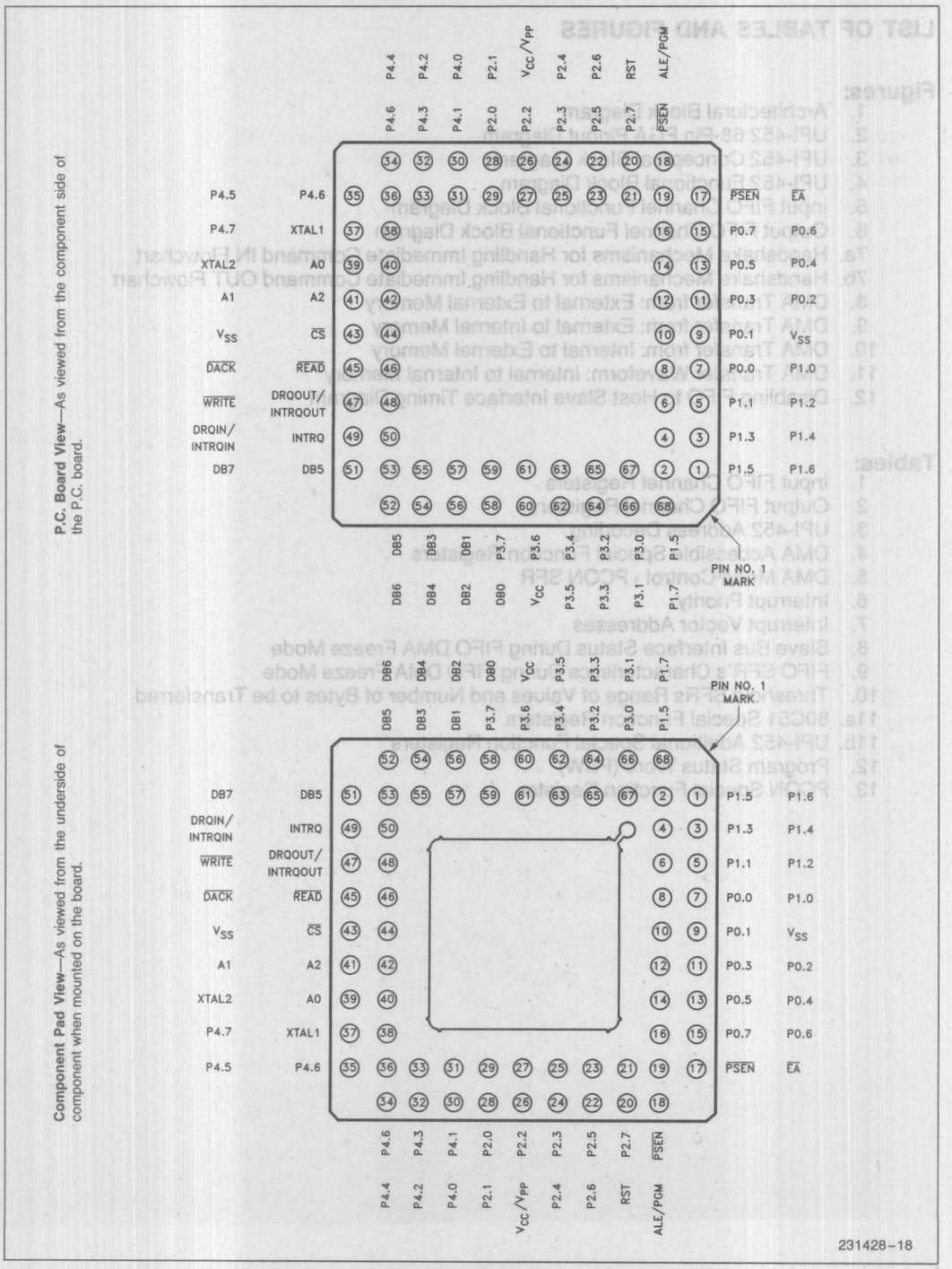


Figure 2. UPI-452 68-Pin PGA Pinout Diagram

UPI MICROCONTROLLER FAMILY

The UPI-452 joins the current members of the UPI microcontroller family. UPI's are derivatives of the MCS™ family of microcontrollers. Because of their on-chip system bus interface, UPI's are designed to be system bus "slaves", while their microcontroller counterparts are intended as system bus "masters".

These UPI Microcontrollers are fully supported by Intel's EPROM programmers (iUP-201) and development tools (ICE, ASM and PLM).

Packaging

The 80C452 comes in a 68-pin PGA (Pin Grid Array) package, while the 87C452P will be offered in a piggyback package. This piggyback package will consist of the standard 68-pin PGA package with a 2764A EPROM soldered on top. These two packages allow designers to use either on-chip EPROM or external memory for their initial designs. The 83C452 (ROM version) will come in the standard 68-pin PGA package. A complete description of 87C452P programming can be found at the end of this data sheet.

UPI Family (Slave Configuration)	MCS Family (Master Configuration)	Speed	RAM (Bytes)	ROM (Bytes)	EPROM (Bytes)
80C452	80C51	12 MHz	256	—	—
83C452	80C51	12 MHz	256	8K	—
87C452P	80C51	12 MHz	256	—	8K
80C452-1	80C51	16 MHz	256	—	—
83C452-1	80C51	16 MHz	256	8K	—
87C452P-1	80C51	16 MHz	256	—	8K

UPI-452 PIN DESCRIPTIONS

Symbol	Pin #	Type	Name and Function
V_{SS}	9/43	I	Circuit Ground.
V_{CC}	60	I	+5V power supply during normal, idle, programming and verification operation. It is also the standby power pin for power down mode.
XTAL1	38	I	Input to the oscillator's high gain amplifier. A crystal or external source can be used.
XTAL2	39	O	Output from the high gain amplifier.
Port 0 (AD0–AD7) P0.0 .1 .2 .3 .4 .5 .6 P0.7	 8 10 11 12 13 14 15 16	I/O	Port 0 is an 8-bit open drain bi-directional I/O port. It is used for data input and output during programming and verification. External pullups are required during program verification. Port 0 can sink eight LS TTL inputs. It is also the multiplexed low-order address and data local expansion bus during accesses to external memory.

UPI-452 PIN DESCRIPTIONS (Continued)

Symbol	Pin #	Type	Name and Function
Port 1 (A0–A7) (HLD, HLDA) P1.0 .1 .2 .3 .4 .5 .6 P1.7	 7 6 5 4 3 2 1 68	I/O	Port 1 is an 8-bit quasi-bi-directional I/O port. It is used for low-order address byte during programming and verification. Port 1 can sink four LS TTL inputs. The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise, the port pin is stuck at 0. Pins P1.5 and P1.6 are multiplexed with HLD and HLDA respectively whose functions are defined as below: Port Pin — Alternate Function P1.5 — $\overline{HLD}$ —Local bus hold input/output signal P1.6 — $\overline{HLDA}$ —Local bus hold acknowledge input
Port 2 (A8–A15) P2.0 .1 .2 .3 .4 .5 .6 .7	 29 28 27 25 24 23 22 21	I/O	Port 2 is an 8-bit quasi-bi-directional I/O port. It also emits the high-order 8 bits of address when accessing local expansion bus external memory (or during 87C452P programming and verification). Port 2 can sink four LS TTL inputs.
Port 3 P3.0 .1 .2 .3 .4 .5 .6 P3.7	 67 66 65 64 63 62 61 59	I/O	Port 3 is an 8-bit quasi-bi-directional I/O port. It is also multiplexed with the interrupt, timer, local serial channel, RD/ and WR/ functions that are used by various options. The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise, the port pin is stuck at 0. Port 3 can sink four LS TTL inputs. The alternate functions assigned to the pins of Port 3 are as follows: Port Pin — Alternate Function P3.0 — RxD — Serial input port P3.1 — TxD — Serial output port P3.2 — INT0 — Interrupt 0 Input P3.3 — INT1 — Interrupt 1 Input P3.4 — T0 — Input to counter 0 P3.5 — T1 — Input to counter 1 P3.6 — WR/ — The write control signal latches the data from Port 0 outputs into the External Data Memory on the local bus. P3.7 — RD/ — The read control signal latches the data from Port 0 outputs on the local bus.

UPI-452 PIN DESCRIPTIONS (Continued)

Symbol	Pin #	Type	Name and Function
Port 4 P4.0 .1 .2 .3 .4 .5 .6 .7	 30 31 32 33 34 35 36 37	I/O	Port 4 is an 8-bit quasi-bi-directional I/O port. Port 4 can sink/source four TTL inputs. It is also used as the control signals during EPROM programming and verification drive pins as follows: Port Pin — Alternate Function P4.5 — '1' during program and verify P4.6 — '0' during program and verify P4.7 — '0' during verify - used as output enable '1' during programming w/ ALE = 0 Note: see Programming and Verification Characteristics in AC/DC Specification section.
RST	20	I	A high level on this pin for two machine cycles while the oscillator is running resets the device. An internal pulldown resistor permits Power-on reset using only a capacitor connected to V_{CC}. This pin does not receive the power down voltage as is the case for HMOS MCS-51 family members. This function has been transferred to the V_{CC} pin.
ALE/PGM	18	I/O	Provides Address Latch Enable output used for latching the address into external memory during normal operation. Receives the program pulse input during EPROM programming. ALE can sink/source eight LS TTL inputs.
$\overline{PSEN}$	19	O	The Program Store Enable output is a control signal that enables the external Program Memory to the bus during normal fetch operation. $\overline{PSEN}$ can sink/source eight LS TTL inputs.
$\overline{EA}$	17	I	When held at TTL high level, the UPI-452 executes instructions from the internal ROM/EPROM when the PC is less than 8192 (8K, 2000H). When held at a TTL low level, the UPI-452 fetches all instructions from external Program Memory.
DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7	58 57 56 55 54 53 52 51	I/O	Host Bus Interface is an 8-bit bi-directional bus. It is used to transfer data and commands between the UPI-452 and the host processor. This bus can sink/source eight LS TTL inputs.
$\overline{CS}$	44	I	This pin is the Chip Select of the UPI-452.
A0 A1 A2	40 41 42	I	These three address lines are used to interface with the host system. They define the UPI-452 operations. The interface is compatible with the Intel microprocessors and the MULTIBUS.
$\overline{READ}$	46	I	This pin is the read strobe from the host CPU. Activating this pin causes the UPI-452 to place the contents of the Output FIFO (either a command or data) or the Host Status/Control Special Function Register on the Slave Data Bus.
$\overline{WRITE}$	47	I	This pin is the write strobe from the host. Activating this pin will cause the value on the Slave Data Bus to be written into the register specified by A0–A2.
DRQIN/ INTRQIN	49	O	This pin requests an input transfer from the host system whenever the Input Channel requires data.
DRQOUT/ INTRQOUT	48	O	This output pin requests an output transfer whenever the Output Channel requires service. If the external host to UPI-452 DMA is enabled, and a Data Stream Command is at the Output FIFO, DRQOUT is deactivated and INTRQ is activated (see 'GENERAL PURPOSE DMA CHANNELS' section).

UPI-452 PIN DESCRIPTIONS (Continued)

Symbol	Pin #	Type	Name and Function
INTRQ	50	O	This output pin is used to interrupt the host processor when an Immediate Command Out or an error condition is encountered. It is also used to interrupt the host processor when the FIFO requests service if the DMA is disabled and INTRQIN and INTRQOUT are not used.
$\overline{DACK}$	45	I	This pin is the DMA acknowledge for the host bus interface Input and Output Channels. When activated, a write command will cause the data on the Slave Data Bus to be written as data to the Input Channel (to the Input FIFO). A read command will cause the Output Channel to output data (from the Output FIFO) on to the Slave Data Bus. This pin should be driven high (+5V) in systems which do not have a DMA controller (see Address Decoding).
V_{CC}/V_{PP}	26	I	+5V power supply during operation. The V_{CC} pin receives the +12V EPROM programming and verification supply voltage.

ARCHITECTURAL OVERVIEW

Introduction

The UPI-452 slave microcontroller incorporates an 80C51 with double the program and data memory, a slave interface which allows it to be connected directly to the host system bus as a peripheral, a FIFO buffer module, a two channel DMA processor, and a fifth I/O port (Figure 3). The UPI-452 retains all of the 80C51 architecture, and is fully compatible with the MCS-51 instruction set.

The Special Function Register (SFR) interface concept introduced in the MCS-51 family of microcontrollers has been expanded in the UPI-452. To the 20 Special Function Registers of the MCS-51, the UPI-452 adds 34 more. These additional Special Function Registers, like those of the MCS-51, provide access to the UPI-452 functional elements including the FIFO, DMA and added interrupt capabilities. Several of the 80C51 core Special Function Registers have also been expanded to support added features of the UPI-452.

This data sheet describes the unique features of the UPI-452. Refer to the 80C51 data sheet for a description of the UPI-452's core CPU functional blocks including;

- Timers/Counters
- I/O Ports
- Interrupt timing and control (other than FIFO and DMA interrupts)
- Serial Channel
- Local Expansion Bus
- Program/Data Memory structure
- Power-Saving Modes of Operation *
- CHMOS Features
- Instruction Set

* except 87C452P piggyback package

Figure 3 contains a conceptual block diagram of the UPI-452. Figure 4 provides a functional block diagram.

FIFO Buffer Interface

A unique feature of the UPI-452 is the incorporation of a 128 byte FIFO array at the host-slave interface. The FIFO allows asynchronous bi-directional transfers between the host CPU and the internal CPU.

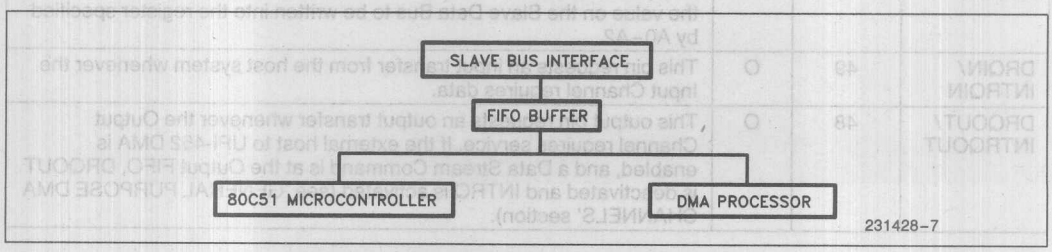


Figure 3. UPI-452 Conceptual Block Diagram

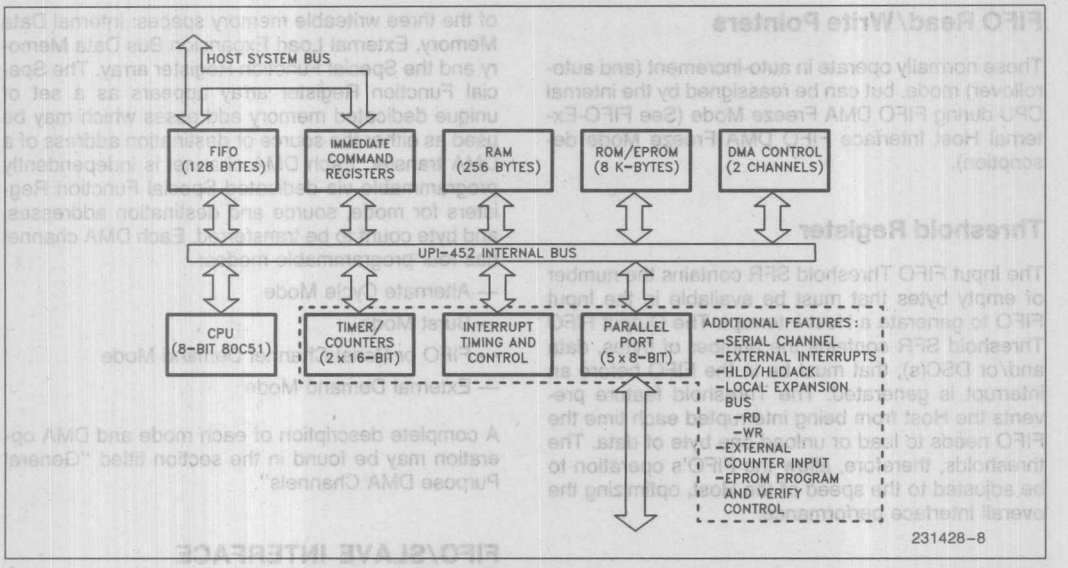


Figure 4. UPI-452 Functional Block Diagram

The division of the 128 bytes between Input and Output channels is user programmable allowing maximum flexibility. If the entire 128 byte FIFO is allocated to the Input channel, a high performance Host can transfer up to 128 bytes at one time, then dedicate its resources to other functions while the internal CPU processes the data in the FIFO. Various handshake signals allow the external Host to operate independently and without frequent monitoring of the UPI-452 internal CPU. The FIFO Buffer insures that the slave processor receives data in the same order that it was sent by the host without the need to keep track of addresses. Three slave bus interface handshake methods are supported by the UPI-452: DMA, Interrupt and Polled.

The FIFO is nine bits wide. The ninth bit acts as a command/data flag. Commands written to the FIFO by either the host or internal CPU are called Data Stream Commands or DSCs. DSCs are written to the input FIFO by the Host via a unique external address. DSCs are written to the output FIFO by the internal CPU via the COMMAND OUT Special Function Register (SFR). When encountered by the host or internal CPU a Data Stream Command can be used as an address vector to user defined service routines. DSCs provide synchronization of data and commands between the Host and internal CPU.

FIFO PROGRAMMABLE FEATURES

Size of Input/Output Channels

The 128 bytes of FIFO space can be allocated between the Input and Output channels via the Channel Boundary Pointer (CBP) SFR. This register contains the number of address locations assigned to the Input channel. The remaining address locations are automatically assigned to the Output FIFO. The CBP SFR can only be programmed by the internal CPU during FIFO DMA Freeze Mode (See FIFO-External Host Interface FIFO DMA Freeze Mode description). The CBP is initialized to 40H (64 bytes) upon reset.

The number in the Channel Boundary Pointer SFR is actually the first address location of the Output FIFO. Writing to the CBP SFR reassigns the Input and Output FIFO address space. Whenever the CBP is written, the Input FIFO pointers are reset to zero and the Output FIFO pointers are set to the value in the CBP SFR.

All of the FIFO space may be assigned to one channel. In such a situation the other channel's data path consists of a single SFR (FIFO IN/COMMAND IN or FIFO OUT/COMMAND OUT SFR) location.

CBP Register	Input FIFO Size	Output FIFO Size
0	1	128
1	1	128
2	2	126
3	3	125
4	4	124
•	•	•
7B	123	5
7C	124	4
7D	125	3
7E	128	1
7F	128	1

FIFO Read/Write Pointers

These normally operate in auto-increment (and auto-rollover) mode, but can be reassigned by the internal CPU during FIFO DMA Freeze Mode (See FIFO-External Host Interface FIFO DMA Freeze Mode description).

Threshold Register

The Input FIFO Threshold SFR contains the number of empty bytes that must be available in the Input FIFO to generate a Host interrupt. The Output FIFO Threshold SFR contains the number of bytes, data and/or DSC(s), that must be in the FIFO before an interrupt is generated. The Threshold feature prevents the Host from being interrupted each time the FIFO needs to load or unload one byte of data. The thresholds, therefore, allow the FIFO's operation to be adjusted to the speed of the Host, optimizing the overall interface performance.

Immediate Commands

The UPI-452 provides, in addition to data and DSCs, a third direct means of communication between the external Host and internal CPU called Immediate Commands. As the name implies, an Immediate Command is available to the receiving CPU immediately, via an interrupt, without being entered into the FIFO as are Data Stream Commands. Like Data Stream Commands, Immediate Commands are written either via a unique external address by the host CPU, or via dedicated SFR by the internal CPU.

The DSC and/or Immediate Command interface may be defined as either Interrupt or Polled under user program control via the Interrupt Enable (IE), Slave Control Register (SLCON), and Interrupt Enable Priority (IEP) Special Function Registers, for the internal CPU and via the Host Control SFR for the external Host CPU.

DMA

The UPI-452 contains a two channel internal DMA controller which allows transfer of data between any of the three writeable memory spaces: Internal Data Memory, External Load Expansion Bus Data Memory and the Special Function Register array. The Special Function Register array appears as a set of unique dedicated memory addresses which may be used as either the source or destination address of a DMA transfer. Each DMA channel is independently programmable via dedicated Special Function Registers for mode, source and destination addresses, and byte count to be transferred. Each DMA channel has four programmable modes:

— Alternate Cycle Mode

— Burst Mode

— FIFO or Serial Channel Demand Mode

— External Demand Mode

A complete description of each mode and DMA operation may be found in the section titled "General Purpose DMA Channels".

FIFO/SLAVE INTERFACE FUNCTIONAL DESCRIPTION

Overview

The FIFO is a 128 Byte RAM array with recirculating pointers to manage the read and write accesses. The FIFO consists of an Input and an Output channel. Access cycles to the FIFO by the internal CPU and external Host are interleaved and appear to be occurring concurrently to both the internal CPU and external Host. Interleaving access cycles ensures efficient use of this shared resource. The internal CPU accesses the FIFO in the same way it would access any of the Special Function Registers e.g., direct and register indirect addressing as well as arithmetic and logical instructions.

Input FIFO Channel

The Input FIFO Channel provides for data transfer from the external Host to the internal CPU (Figure 5). The registers associated with the Input Channel during normal operation are listed in Table 1*.

Table 1. Input FIFO Channel Registers*

	Register Name	Description
1)	Input Buffer Latch	Host CPU Write only
2)	FIFO IN SFR	Internal CPU Read only
3)	COMMAND IN SFR	Internal CPU Read only
4)	Input FIFO Read Pointer SFR	Internal CPU Read only
5)	Input FIFO Write Pointer SFR	Internal CPU Read only
6)	Input FIFO Threshold SFR	Internal CPU Read only

*See "FIFO-EXTERNAL HOST INTERFACE FIFO DMA FREEZE MODE" section for FIFO DMA Freeze Mode SFR characteristics description.

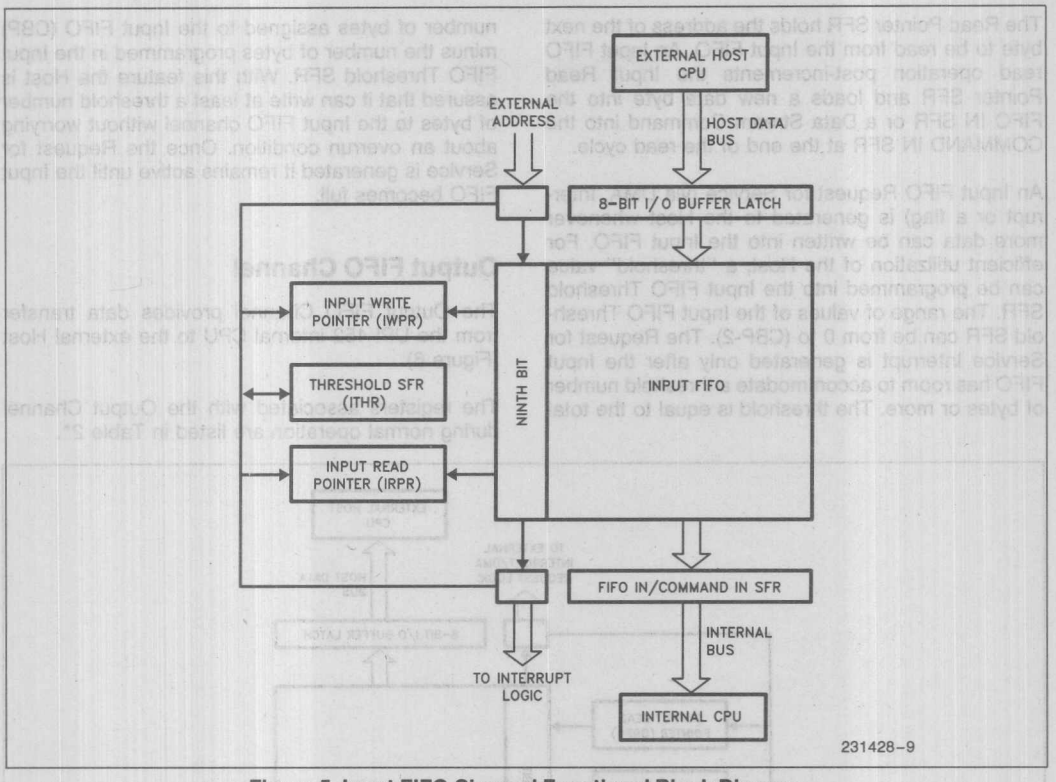


Figure 5. Input FIFO Channel Functional Block Diagram

The host CPU writes data and Data Stream Commands into the Input Buffer Latch on the rising edge of the external WR signal. External addressing determines whether the byte is a data byte or Data Stream Command and the FIFO logic sets the ninth bit of the FIFO accordingly as the byte is moved from the Input Buffer Latch into the FIFO. A "1" in the ninth bit indicates that the incoming byte is a Data Stream Command. The internal CPU reads data bytes via the FIFO IN SFR, and Data Stream Commands via the COMMAND IN SFR.

A Data Stream Command will generate an interrupt to the internal CPU prior to being read and after completion of the previous operation. The DSC can then be read via the COMMAND IN SFR. Data can only be read via the FIFO IN SFR and Data Stream Commands via the COMMAND IN SFR. Attempting to read Data Stream Commands as data by addressing the FIFO IN SFR will result in "0FFH" being read, and the Input FIFO Read Pointer will remain intact. (This prevents accidental misreading of Data Stream Commands.) Attempting to read data as Data Stream Commands will have the same consequence.

The Input FIFO Channel addressing is controlled by the Input FIFO Read and Write Pointer SFRs. These SFRs are read only registers during normal operation. However, during FIFO DMA Freeze Mode (See FIFO-External Host Interface FIFO DMA Freeze Mode description), the internal CPU has write access to them. Any write to these registers in normal mode will have no effect. The Input Write Pointer SFR contains the address location to which data/commands are written from the Input Buffer Latch. The write pointer is automatically incremented after each write and is reset to zero if equal to the CBP, as the Input FIFO operates as a circular buffer.

If a write is performed on an empty FIFO, the first byte is also written into the FIFO IN or COMMAND IN SFR. If the Host continues writing while the Input FIFO is full, an external interrupt, if enabled, is sent to the host to signal the overrun condition. The writes are ignored by the FIFO control logic. Similarly, an internal CPU read of an empty FIFO will cause an underrun error interrupt to be generated to the internal CPU and a value of "0FFH" will be read by the internal CPU.

The Read Pointer SFR holds the address of the next byte to be read from the Input FIFO. An Input FIFO read operation post-increments the Input Read Pointer SFR and loads a new data byte into the FIFO IN SFR or a Data Stream Command into the COMMAND IN SFR at the end of the read cycle.

An Input FIFO Request for Service (via DMA, Interrupt or a flag) is generated to the Host whenever more data can be written into the Input FIFO. For efficient utilization of the Host, a "threshold" value can be programmed into the Input FIFO Threshold SFR. The range of values of the Input FIFO Threshold SFR can be from 0 to (CBP-2). The Request for Service Interrupt is generated only after the Input FIFO has room to accommodate a threshold number of bytes or more. The threshold is equal to the total number of bytes assigned to the Input FIFO (CBP) minus the number of bytes programmed in the Input FIFO Threshold SFR. With this feature the Host is assured that it can write at least a threshold number of bytes to the Input FIFO channel without worrying about an overrun condition. Once the Request for Service is generated it remains active until the Input FIFO becomes full.

Output FIFO Channel

The Output FIFO Channel provides data transfer from the UPI-452 internal CPU to the external Host (Figure 6).

The registers associated with the Output Channel during normal operation are listed in Table 2*.

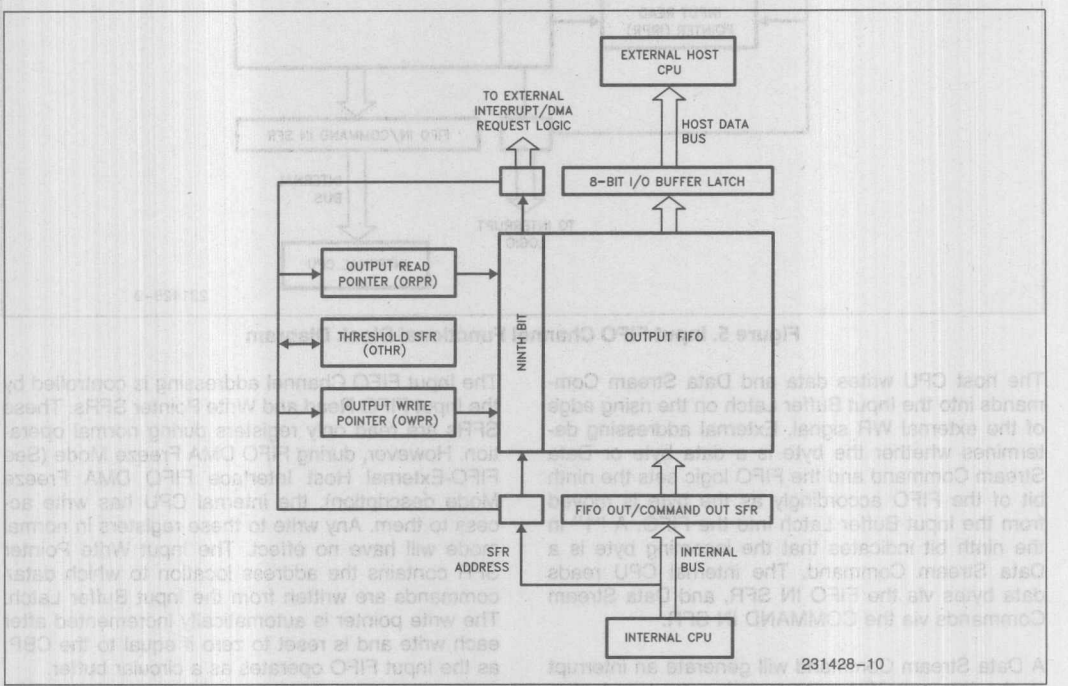


Figure 6. Output FIFO Channel Functional Block Diagram

Table 2. Output FIFO Channel Registers

	Register Name	Description
1)	Output Buffer Latch	Host CPU Read only
2)	FIFO OUT SFR	Internal CPU Read and Write
3)	COMMAND OUT SFR	Internal CPU Read and Write
4)	Output FIFO Read Pointer SFR	Internal CPU Read only
5)	Output FIFO Write Pointer SFR	Internal CPU Read only
6)	Output FIFO Threshold SFR	Internal CPU Read only

*See "FIFO-EXTERNAL HOST INTERFACE FIFO DMA FREEZE MODE" section for FIFO DMA Freeze Mode register characteristics description.

The UPI-452 internal CPU transfers data to the Output FIFO via the FIFO OUT SFR and commands via the COMMAND OUT SFR. If the byte is written to the COMMAND OUT SFR, the ninth bit is automatically set (=1) to indicate a Data Stream Command. If the byte is written to the FIFO OUT SFR the ninth bit is cleared (=0). Thus the FIFO OUT and COMMAND OUT SFRs are the same but the address determines whether the byte entered in the FIFO is a DSC or data byte.

The Output FIFO preloads a byte into the Output Buffer Latch. When the Host issues a RD/ signal, the data is immediately read from the Output Buffer Latch. The next data byte is then loaded into the Output Buffer Latch, a flag is set and an interrupt, if enabled, is generated if the byte is a DSC (ninth bit is set). The operation is carefully timed such that an interrupt can be generated in time for it to be recognized by the Host before its next read instruction. Internal CPU write and external Host read operations are interleaved at the FIFO so that they appear to be occurring concurrently.

The Output FIFO read and write pointer operation is the same as for the Input Channel. Writing to the FIFO OUT or COMMAND OUT SFRs will increment the Output Write Pointer SFR but reading from it will leave the write pointer unchanged. A rollover of the Output FIFO Write Pointer causes the pointer to be reset to the value in the Channel Boundary Pointer (CBP) SFR.

If the external host attempts to read a Data Stream Command as a data byte it will result in invalid data (0FFH) being read. The DSC is not lost because the invalid read does not increment the pointer. Similarly attempting to read a data byte as a Data Stream Command has the same result.

A Request for Service is generated to the external Host under the following two conditions:

1.) Whenever the internal CPU has written a threshold number of bytes or more into the Output FIFO (threshold = (OTHR) + 1). The threshold number should be chosen such that the bus latency time for the external Host does not result in a FIFO overrun error condition on the internal CPU side. The threshold limit should be large enough to make a bus request by the UPI-452 to the external host CPU worthwhile. Once a request for service is generated, the request remains active until the Output FIFO becomes empty. The range of values of the FIFO Output Threshold (OTHR) SFR is from 1 to the Output FIFO Size. The threshold number can be programmed via the OTHR SFR.

2.) The second type of Request for Service is called "Flush Mode" and occurs when the internal CPU writes a Data Stream Command into the Output FIFO. Its purpose is to ensure that a data block entered into the Output FIFO, which is less than the programmed threshold, will generate a Request for Service interrupt, if enabled, and be read, or "Flushed" from the Output FIFO, by the external host CPU regardless of the status of the OTHR SFR.

Immediate Commands

Immediate Commands provide direct communication between the external Host and UPI-452. Unlike Data Stream Commands which are entered into the FIFO, the Immediate Command is available to the receiving CPU directly, bypassing the FIFO. The Immediate Command can serve as a program vector pointing into a jump table in the recipients software. Immediate Command Interrupts are generated, if enabled, and a bit in the appropriate Status Register is set when an Immediate Command is input or output. A similar bit is provided to acknowledge when an Immediate Command has been read and whether the register is available to receive another command. The bits are reset when the Immediate Commands are read. Two Special Function Registers are dedicated to the Immediate Command interface. External addressing determines whether the Host is accessing the Input FIFO or the Immediate Command IN (IMIN) SFR. The internal CPU writes Immediate Commands to the Immediate Command OUT (IMOUT) SFR.

Both processors have the ability to enable or disable Immediate Command Interrupts. By disabling the interrupt, the recipient of the Immediate Command can poll the status SFR and read the Immediate Command at its convenience. Immediate Commands should only be written when the appropriate Immediate Command SFR is empty (as indicated in the appropriate status SFR:HSTAT/SSTAT). Similarly, the Immediate Command SFR should only be read when there is data in the Register.

The flowcharts in Figure 7a and 7b illustrate the proper handshake mechanisms between the external Host and internal CPU when handling Immediate Commands.

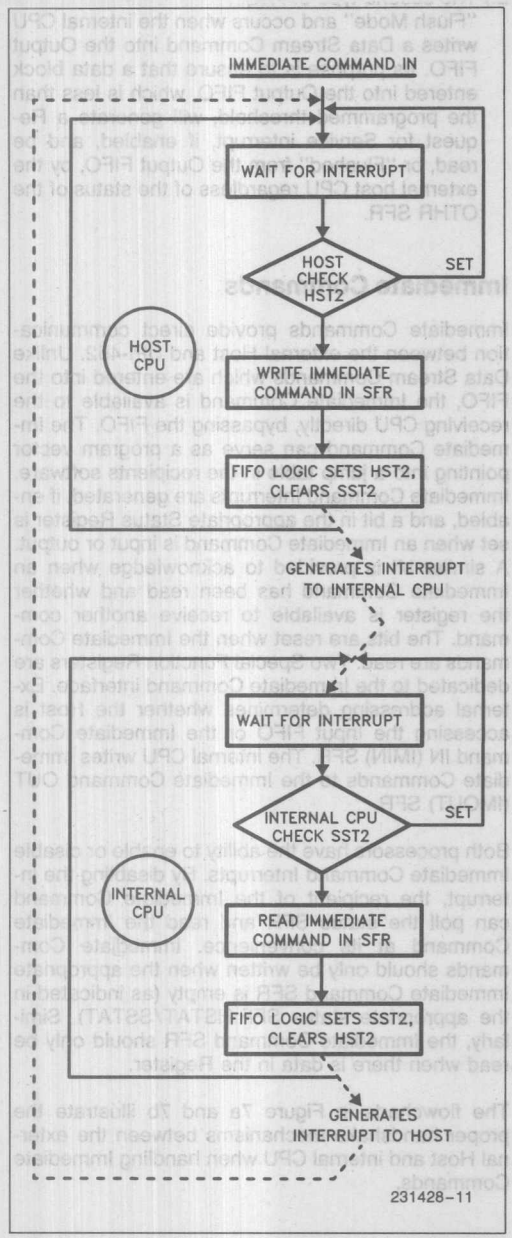


Figure 7a. Handshake Mechanisms for Handling Immediate Command IN Flowchart

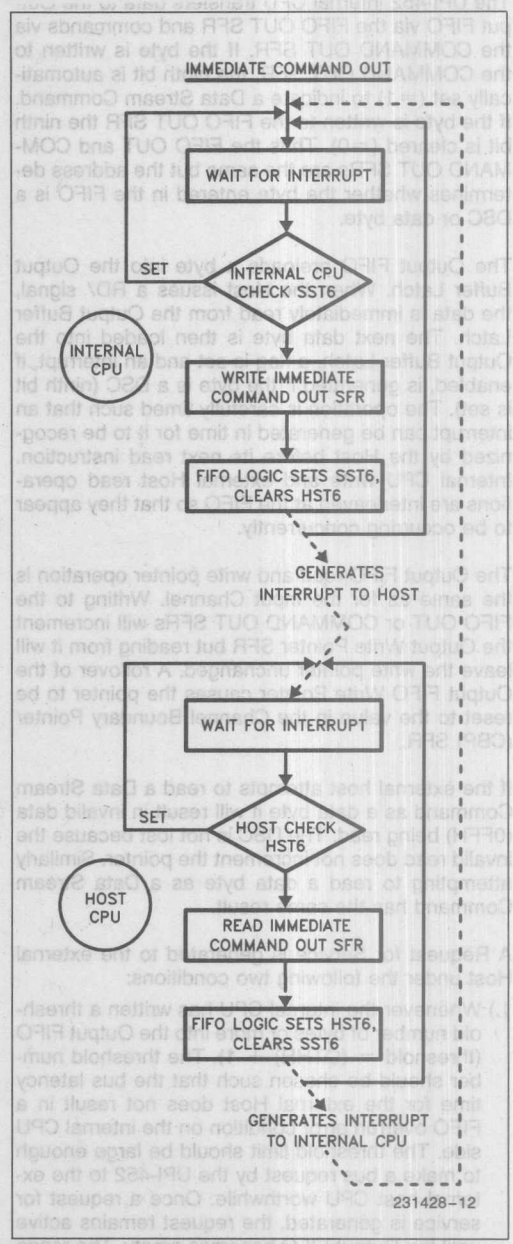


Figure 7b. Handshake Mechanisms for Handling Immediate Command OUT Flowchart

HOST & SLAVE INTERFACE SPECIAL FUNCTION REGISTERS

Slave Interface Special Function Registers

The Internal CPU interfaces with the FIFO slave module via the following registers:

1) Mode Special Function Register (MODE)
2) Slave Control Special Function Register (SLCON)
3) Slave Status Special Function Register (SSTAT)

Each register resides in the SFR Array and is accessible via all direct addressing modes except bit. Only the Slave Control Register (SLCON) is bit addressable.

1) MODE Special Function Register (MODE)

The MODE SFR provides the primary control of the external host-FIFO interface. It is included in the SFR Array so that the internal CPU can configure the external host-FIFO interface should the user decide that the UPI-452 slave initialize itself independent of the external host CPU.

The MODE SFR can be directly modified by the internal CPU through direct address instructions. It can also be indirectly modified by the external host CPU by setting up a MODE SFR service routine in the UPI-452 program memory and having the host issue a Command, either Immediate or DSC, to vector to that routine.

Symbolic Address									Physical Address
MODE	—	MD6	MD5	MD4	—	—	—	—	0F9H
	(MSB)							(LSB)	
	Status On Reset:								
	1*	0	0	0	1*	1*	1*	1*	

MD7 (reserved)**

MD6 Request for Service to external CPU via;

1 = DMA (DRQIN/DRQOUT) request to external host when the Input or Output FIFO channel requests service

0 = Interrupt (INTRQIN/INTRQOUT or INTRQ) to external host when the Input or Output FIFO channel requests service or a DSC is encountered in the I/O Buffer Latch

MD5 Configure DRQIN/INTRQIN and DRQOUT/INTRQOUT to be either;

1 = Enable (Actively driven)

0 = Disable (Tri-state)

MD4 Configure INTRQ to be either;

1 = Enable (Actively driven)

0 = Disable (Tri-state)

MD3 (reserved) **

MD2 (reserved) **

MD1 (reserved) **

MD0 (reserved) **

2) Slave Control SFR (SLCON)

The Slave Control SFR is used to configure the FIFO-internal CPU interface. All interrupts are to the internal CPU.

Symbolic Address									Physical Address
SLCON	IFI	OFI	ICII	ICOI	FRZ	—	IFRS	OFRS	0E8H
	(MSB)							(LSB)	
Status On Reset:	0	0	0	0	0	1*	0	0	

IFI Enable Input FIFO Interrupt (due to Underrun Error Condition, Data Stream Command or Request Service)
1 = Enable
0 = Disable

OFI Enable Output FIFO Interrupt (due to Overrun Error Condition or Request Service)
1 = Enable
0 = Disable
Note: If the DMA is configured to service a FIFO demand, then the Request for Service Interrupt is not generated.

ICII Generate Interrupt when a command is written to the Immediate Command in Register
1 = Enable
0 = Disable

ICOI Generate Interrupt when Immediate Command Out Register is Available
1 = Enable
0 = Disable

FRZ Enable FIFO DMA Freeze Mode
1 = Normal operation
0 = FIFO DMA Freeze Mode

SC2 (reserved) **

IFRS Input FIFO Channel Request for Service
1 = Request when Input FIFO not empty
0 = Request when Input FIFO full

OFRS Output FIFO Channel Request for Service
1 = Request when Output FIFO not full
0 = Channel Request when Output FIFO empty

NOTES:
*A '1' will be read from all SFR reserved locations except HCON SFR, HC0 and HC2.
**'reserved'—these locations are reserved for future use by Intel Corporation.

3) Slave Status SFR (SSTAT)

The bits in the Slave Status SFR reflect the status of the FIFO-internal CPU interface. It can be read during an internal interrupt service routine to determine the nature of the interrupt or read during a polling sequence to determine a course of action.

Symbolic Address									Physical Address
SSTAT	SST7	SST6	SST5	SST4	SST3	SST2	SST1	SST0	0E9H
	← Output FIFO Status →				← Input FIFO Status →				
Status On Reset:	1	0	0	0	1	1	1	1	
	(MSB)							(LSB)	

SST7 Output FIFO Overrun Error Condition
- 1 = No Error
- 0 = Error (latched until Slave Status SFR is read)

SST6 Immediate Command Out Register Status
- 1 = Full (i.e. Host CPU has not read previous Immediate Command Out sent by internal CPU)
- 0 = Available

SST5 FIFO DMA Freeze Mode Status
- 1 = Normal Operation
- 0 = FIFO DMA Freeze Mode in Progress

SST4 Output FIFO Request for Service Flag
- 1 = Output FIFO does not request service
- 0 = Output FIFO requests service

SST3 Input FIFO Underrun Error Condition Flag
- 1 = No Underrun Error
- 0 = Underrun Error (latched until Slave Status SFR is read)

SST2 Immediate Command In SFR Status
- 1 = Empty
- 0 = Immediate Command received from host CPU

SST1 Data Stream Command/Data at Input FIFO Flag
- 1 = Data (not DSC)
- 0 = DSC (at COMMAND IN SFR)

SST0 Input FIFO Request For Service Flag
- 1 = Input FIFO Does Not Request Service
- 0 = Input FIFO Request for Service

EXTERNAL HOST INTERFACE SPECIAL FUNCTION REGISTERS

The external host CPU has direct access to the following SFRs:

1) Host Control Special Function Register
2) Host Status Special Function Register

It can also access other SFRs by commanding the internal CPU to change them accordingly via Data Stream Commands or Immediate Commands. The protocol for implementing this is entirely determined by the user.

1) Host Control SFR (HCON)

By writing to the Host Control SFR, the host can enable or disable FIFO interrupts and DMA requests and can reset the UPI-452.

Symbolic Address									Physical Address
HCON	HC7	HC6	HC5	HC4	HC3	—	HC1	—	0E7H
	(MSB)							(LSB)	
	Status On Reset:								
	0	0	0	0	0	0*	0	0*	

HC7 Enable Output FIFO Interrupt due to Underrun Error Condition, Data Stream Command or Service Request
- 1 = Enable
- 0 = Disable

HC6 Enable Input FIFO Interrupt due to Overrun Error Condition, or Service Request
- 1 = Enable
- 0 = Disable

HC5 Enable the generation of the Interrupt due to Immediate Command Out being present
- 1 = Enable
- 0 = Disable

HC4 Enable the Interrupt due to the Immediate Command In Register being Available for a new Immediate Command byte
- 1 = Enable
- 0 = Disable

HC3 Reset UPI-452
- 1 = Software RESET
- 0 = Normal Operation

HC2 (reserved) **

HC1 Select between INTRQ and INTRQIN/INTRQOUT as Request for Service interrupt signal when DMA is disabled
- 1 = INTRQ
- 0 = INTRQIN or INTRQOUT

HC0 (reserved) **

NOTES:
*A '1' will be read from all SFR reserved locations except HCON SFR, HC0 and HC2.
**'reserved'—these locations are reserved for future use by Intel Corporation.

2) Host Status SFR (HSTAT)

The Host Status SFR provides information on the FIFO-Host Interface and can be used to determine the source of an external interrupt during polling. Like the Slave Status SFR, the Host Status SFR reflects the current status of the FIFO-external host interface.

Symbolic Address									Physical Address
HSTAT	HST7	HST6	HST5	HST4	HST3	HST2	HST1	HST0	0E6H
	← Output FIFO Status →				← Input FIFO Status →				
Status On Reset:									
	1	1	1	1	1	1/0*	1	1	
	(MSB)							(LSB)	

HST7 Output FIFO Underrun Error Condition
1 = No Underrun Error
0 = Underrun Error (latched until Host Status Register is read)

HST6 Immediate Command Out SFR Status
1 = Empty
0 = Immediate Command Present

HST5 Data Stream Command/Data at Output FIFO Status
1 = Data (not DSC)
0 = DSC (present at Output FIFO COMMAND OUT SFR)
(Note: Only if HST4 = 0, if HST4 = 1 then undetermined)

HST4 Output FIFO Request for Service Status
1 = No Request for Service
0 = Output FIFO Request for Service due to:

a. Output FIFO containing the threshold number of bytes or more

b. Internal CPU sending a block of data terminated by a DSC (DSC Flush Mode)

HST3 Input FIFO Overrun Error Condition
1 = No Overrun Error
0 = Overrun Error (latched until Host Status Register is read)

HST2 Immediate Command In SFR Status
1 = Full (i.e. Internal CPU has not read previous Immediate Command sent by Host)
0 = Empty

* Reset value;

'1' — if read by the external Host

'0' — if read by internal CPU (reads shadow latch - see FIFO DMA Freeze Mode description)

HST1 FIFO DMA Freeze Mode Status
1 = Freeze Mode in progress.
(In Freeze Mode, the bits of the Host Status SFR are forced to a '1' initially to prevent the external Host from attempting to access the FIFO. The definition of the Host Status SFR bits during FIFO DMA Freeze Mode can be found in FIFO DMA Freeze Mode description)
0 = Normal Operation

HST0 Input FIFO Request Service Status
1 = Input FIFO does not request service
0 = Input FIFO request service due to the Input FIFO containing enough space for the host to write the threshold number of bytes or more

FIFO MODULE - EXTERNAL HOST INTERFACE

Overview

The FIFO-external Host interface supports high speed asynchronous bi-directional 8-bit data transfers. The host interface is fully compatible with Intel microprocessor local busses and with MULTIBUS. The FIFO has two specialized DMA request pins for Input and Output FIFO channel DMA requests. These are multiplexed to provide a dedicated Request for Service interrupt (DRQIN/INTRQIN, DRQOUT/INTRQOUT).

The external Host can program, under user defined protocol, thresholds into the FIFO Input and Output Threshold SFRs which determine when the FIFO Request for Service interrupt is generated to the Host CPU. The FIFO module external Host interface is configured by the internal CPU via the MODE SFR. "The external Host can enable and disable Host interface interrupts via the Host Control SFR." Data Stream Commands in the Input FIFO channel allow the Host to influence the processing of data blocks and are sent with the data flow to maintain synchronization. Data Stream Commands in the Output FIFO Channel allow the internal CPU to perform the same function, and also to set the Output FIFO Request Service status logic to the host CPU regardless of the programmed value in the Threshold SFR.

Slave Interface Address Decoding

The UPI-452 determines the desired Host function through address decoding. The lower three bits of the address as well as the READ, WRITE, Chip Select ($\overline{CS}$) and DMA Acknowledge ($\overline{DACK}$) are used for decoding. Table 3 shows the pin states and the Read or Write operations associated with each configuration.

Interrupts to the Host

The UPI-452 interrupts the external Host via the INTRQ pin. In addition, the DRQIN and DRQOUT pins can be multiplexed as interrupt request lines, INTRQIN and INTRQOUT respectively, when DMA is disabled. This provides two special FIFO "Request for Service" interrupts.

There are eight FIFO-related interrupt sources; two from The Input FIFO; three from The Output FIFO; one from the Immediate Command Out SFR; one from the Immediate Command IN SFR; and one due to FIFO DMA Freeze Mode.

INPUT FIFO: The Input FIFO interrupt is generated whenever:

a. The Input FIFO contains space for a threshold number of bytes.

Table 3. UPI-452 Address Decoding

$\overline{DACK}$	$\overline{CS}$	A2	A1	A0	Read	Write
1	1	X	X	X	No Operation	No Operation
1	0	0	0	0	Data or DMA from Output FIFO Channel	Data or DMA to Input FIFO Channel
1	0	0	0	1	Data Stream Command from Output FIFO Channel	Data Stream Command to Input FIFO Channel
1	0	0	1	0	Host Status SFR Read	Reserved
1	0	0	1	1	Host Control SFR Read	Host Control SFR Write
1	0	1	0	0	Immediate Command SFR Read	Immediate Command to SFR Write
1	0	1	1	X	Reserved	Reserved
0	X	X	X	X	DMA Data from Output FIFO Channel	DMA Data to Input FIFO Channel
1	0	1	0	1	Reserved	Reserved

NOTES:
1. Attempting to read a DSC as a data byte will result in invalid data being read. The read pointers are not incremented so that the DSC is not lost. Attempting to read a data byte as a DSC has the same result.
2. If $\overline{DACK}$ is active the UPI-452 will attempt a DMA operation when $\overline{RD}$ or $\overline{WR}$ becomes active regardless of the DMA enable bit (MD6) in the MODE SFR. Care should be taken when using $\overline{DACK}$. For proper operation, $\overline{DACK}$ must be driven high (+5V) when not using DMA.

b. When an Input FIFO overrun error condition exists. The appropriate bits in the Host Status SFR are set and the interrupt is generated only if enabled.

OUTPUT FIFO: The Output FIFO Request for Service Interrupt operates in a similar manner as the Input FIFO interrupt:

a. When the FIFO contains the threshold number of bytes or more.

b. Output FIFO error condition interrupts are generated when the Output FIFO is underrun.

c. Data Stream Command present in the Output Buffer Latch.

A Data Stream Command interrupt is used to halt normal processing, using the command as a vector to a service routine. When DMA is disabled, the user may program (through HC1) INTRQ to include FIFO Request for Service Interrupts or use INTRQIN and INTRQOUT as Request for Service Interrupts.

IMMEDIATE COMMAND INTERRUPTS:

a. An Immediate Command Out Interrupt is generated, if enabled, to the Host and the corresponding Host Status SFR bit (HSTAT HST6) is cleared, when the internal CPU writes to the Immediate Command OUT (IMOUT) SFR. When the Host reads the Immediate Command OUT (IMOUT) SFR the corresponding bit in the Host Status (HSTAT) SFR is set. This causes the Slave Status Immediate Command OUT Status bit (SSTAT SST6) to be cleared indicating that the Immediate Command OUT (IMOUT) SFR is empty. If enabled, a FIFO-Slave Interface will also be generated to the internal CPU. (See Figure 7b, Immediate Command OUT Flowchart.)

b. An Immediate Command IN interrupt is generated, if enabled, to the Host when the internal CPU has read a byte from the Immediate Command IN (IMIN) SFR. The read operation clears the Host Status SFR Immediate Command IN Status bit (HSTAT HST2) indicating that the Immediate Command IN SFR is empty. The corresponding Slave Status (SSTAT) SFR bit is also set to indicate an empty status. Setting the Slave Status SFR bit generates a FIFO-Slave Interface interrupt, if enabled, to the internal CPU. (See Figure 7a, Immediate Command IN Flowchart.)

NOTE:
Immediate Command IN and OUT interrupts are actually specific Request For Service interrupts to the Host.

FIFO DMA FREEZE MODE: When the internal CPU invokes FIFO DMA Freeze Mode, for example at reset or to reconfigure the FIFO interface, INTRQ is activated. The INTRQ can only be deactivated by the external Host reading the Host Status SFR (HST1 remains active until FIFO DMA Freeze Mode is disabled by the internal CPU).

Once an interrupt is generated, INTRQ will remain high until no interrupt generating condition exists. For a FIFO underrun/overrun error interrupt, the interrupt condition is deactivated by the external Host reading the Host Status SFR. An interrupt is serviced by reading the Host Status SFR to determine the source of the interrupt and vectoring the appropriate service routine.

DMA Requests to the Host

The UPI-452 generates two DMA requests, DRQIN and DRQOUT, to facilitate data transfer between the Host and the Input and Output FIFO channels. A DMA acknowledge, $\overline{DACK}$, is used as a chip select and initiates a data transfer. The external $\overline{READ}$ and $\overline{WRITE}$ signals select the Input and Output FIFO respectively. The $\overline{CS}$ and address lines can also be used as a DMA acknowledge for processors with onboard DMA controllers which do not generate a $\overline{DACK}$ signal.

The internal CPU can configure the UPI-452 to request service from the external host via DMA or interrupts by programming Mode SFR MD6 bit. In addition the external Host enables DMA requests through bits 6 and 7 of the Host Control SFR. When a DMA request is invoked the number of bytes transferred to the Input FIFO is the total number of bytes in the Input FIFO (as determined by the CBP SFR) minus the value programmed in the Input FIFO Threshold SFR. The DMA request line is activated only when the Input FIFO has a threshold number of bytes that can be transferred.

The Output FIFO DMA request is activated when a DSC is written by the internal CPU at the end of a less than threshold size block of data (Flush Mode) or when the Output FIFO threshold is reached. The request remains active until the Input FIFO becomes full or the Output FIFO becomes empty. If a DSC is encountered during an Output FIFO DMA transfer, the DMA request is dropped until the DSC is read. The DMA request will be reactivated after the DSC is read and remains active until the Output FIFO becomes empty or another DSC is encountered.

FIFO MODULE - INTERNAL CPU INTERFACE

Overview

The Input and Output FIFOs are accessed by the internal CPU through direct addressing of the FIFO IN/COMMAND IN and FIFO OUT/COMMAND OUT Special Function Registers. All of the 80C51 instructions involving direct addressing may be used to access the FIFO's SFRs. The FIFO IN, COMMAND IN and Immediate Command In SFRs are actually read only registers, and their Output counterparts are write only. Internal DMA transfers data between Internal memory, External Memory and the Special Function Registers. The Special Function Registers appear as another group of dedicated memory addresses and are programmed as the source or destination via the DMA0/DMA1 Source Address or Destination Address Special Function Registers. The FIFO module manages the transfer of data between the external host and FIFO SFRs.

Internal CPU Access to FIFO Via Software Instructions

The internal CPU has access to the Input and Output FIFOs via the FIFO IN/COMMAND IN and FIFO OUT/COMMAND OUT SFRs which reside in the Special Function Register Array. At the end of every instruction that involves a read of the FIFO IN/COMMAND IN SFR, the SFR is written over by a new byte from the Input FIFO channel when available. At the end of every instruction that involves a write to the FIFO OUT/COMMAND OUT SFR, the new byte is written into the Output FIFO channel and the write pointer is incremented after the write operation (post incremented).

The internal CPU reads the Input FIFO by using the FIFO IN/COMMAND IN SFR as the source register in an instruction. Those instructions which read the Input FIFO are listed below:

```
ADD  A,FIFO IN/COMMAND IN
ADDC A,FIFO IN/COMMAND IN
PUSH FIFO IN/COMMAND IN
ANL  A,FIFO IN/COMMAND IN
ORL  A,FIFO IN/COMMAND IN
XRL  A,FIFO IN/COMMAND IN
CJNE A,FIFO IN/COMMAND IN, rel
SUBB A,FIFO IN/COMMAND IN
MOV  direct,FIFO IN/COMMAND IN
MOV  @Ri,FIFO IN/COMMAND IN
MOV  Rn,FIFO IN/COMMAND IN
MOV  A,FIFO IN/COMMAND IN
```

After each access to these registers, they are overwritten by a new byte from the FIFO.

NOTE:
Instructions which use the FIFO IN or COMMAND IN SFR as both a source and destination register will have the data destroyed as the next data byte is rewritten into the FIFO IN register at the end of the instruction. These instructions are not supported by the UPI-452 FIFO. Data can only be read through the FIFO IN SFR and DSCs through the COMMAND IN SFR. Data read through the COMMAND IN SFR will be read as 0FFH, and DSCs read through the FIFO IN SFR will be read as OFFH. The Immediate Command in SFR is read with the same instructions as the FIFO IN and COMMAND IN SFRs.

The FIFO IN, COMMAND IN and Immediate Command In SFRs are read only registers. Any write operation performed on these registers will be ignored and the FIFO pointers will remain intact.

The internal CPU uses the FIFO OUT SFR to write to the Output FIFO and any instruction which uses the FIFO OUT or COMMAND OUT SFR as a destination will invoke a FIFO write. DSCs are differentiated from data by writing to the COMMAND OUT SFR. In the FIFO, Data Stream Commands have the ninth bit associated with the command byte set to "1". The instructions used to write to the Output FIFO are listed below:

```
MOV FIFO OUT/COMMOUT, A
MOV FIFO OUT/COMMOUT, direct
MOV FIFO OUT/COMMOUT, Rn
POP FIFO OUT/COMMOUT
MOV FIFO OUT/COMMOUT, #data
MOV FIFO OUT/COMMOUNT, @Ri
```

NOTE:
Instructions which use the FIFO OUT/COMMAND OUT SFRs as both a source and destination register cause invalid data to be written into the Output FIFO. These instructions are not supported by the UPI-452 FIFO.

GENERAL PURPOSE DMA CHANNELS

Overview

There are two identical General Purpose DMA Channels on the UPI-452 which allow high speed data transfer from one writeable memory space to another. As many as 64K bytes can be transferred in a single DMA operation. The following memory spaces can be used with DMA channels:

- Internal Data Memory
- External Data Memory
- Special Function Registers

The Special Function Register array appears as a limited group of dedicated memory addresses. The Special Function Registers may be used in DMA transfer operations by specifying the SFR as the source or destination address. The Special Function Registers which may be used in DMA transfers are listed in Table 4. Table 4 also shows whether the SFR may be used as Source or Destination only, or both.

The FIFO can be accessed during DMA by using the FIFO IN SFR as the DMA Source Address Register (SAR) or the FIFO OUT SFR as the Destination Address Register (DAR). (Note: Since the FIFO IN SFR is a read only register, the DMA transfer will be ignored if it is used as a DMA DAR. This is also true if the FIFO OUT SFR is used as a DMA SAR.)

Each DMA channel is software programmable to operate in either Block Mode or Demand Mode. In the Block Mode, DMA transfers can be further programmed to take place in Burst Mode or Alternate Cycle mode. In Burst Mode, the processor halts its execution and dedicates its resources to the DMA transfer. In Alternate Cycle Mode, DMA cycles and instruction cycles occur alternately.

In Demand Mode, a DMA transfer occurs only when it is demanded. Demands can be accepted from an external device (through External Interrupt pins, EXT0/EXT1) or from either the Serial Channel or FIFO flags. In this way, a DMA transfer can be synchronized to an external device, the FIFO or the Serial Port. If the External Interrupt is configured in Edge Mode, a single byte transfer occurs per transition. The external interrupt itself will occur if enabled. If the External Interrupt is configured in Level Mode, DMA transfers continue until the External Interrupt request goes inactive or the byte count becomes zero. The following flags activate Demand Mode transfers of one byte to/from the FIFO or Serial Channel:

RI - Serial Channel Receiver Buffer Full

TI - Serial Channel Transmitter Buffer Empty

Architecture

There are three 16 bit and one 8 bit Special Function Registers associated with each DMA channel.

- The 16 bit Source Address SFR (SAR) points to the source byte.
- The 16 bit Destination Address SFR (DAR) points to the destination.
- The 16 bit Byte Count SFR (BCR) contains the number of bytes to be transferred and is decremented when a byte transfer is accomplished.
- The DMA Control SFR (DCON) is eight bits wide and specifies the source memory space, destination memory space and the mode of operation.

In Auto Increment mode, the Source Address and/or Destination Address is incremented when a byte is transferred. When a DMA transfer is complete (BCR = 0), the DONE bit is set and a maskable interrupt is generated. The GO bit must be set to start any DMA transfer (also, the Slave Control SFR FRZ bit must be set to disable FIFO DMA Freeze Mode). The two DMA channels are designated as DMA0 and DMA1, and their corresponding registers are suffixed by 0 or 1; e.g. SAR0, DAR1, etc.

Table 4. DMA Accessible Special Function Registers

SFR	Symbol	Address	Source Only	Destination Only	Either
Accumulator	A/ACC	0E0H			Y
B Register	B	0F0H			Y
FIFO IN	FIN	0EEH	Y		
COMMAND IN	CIN	0EFH	Y		
FIFO OUT	FOUT	0FEH		Y	
COMMAND OUT	COUT	0FFH		Y	
Serial Data Buffer	SBUF	099H			Y
Port 0	P0	080H			Y
Port 1	P1	090H			Y
Port 2	P2	0A0H			Y
Port 3	P3	0B0H			Y
Port 4	P4	0C0H			Y

DMA Special Function Registers

DMA Control SFR: DCON0, DCON1

Symbolic Address									Physical Address
DCON0	DAS	IDA	SAS	ISA	DM	TM	DONE	GO	092H
DCON1	DAS	IDA	SAS	ISA	DM	TM	DONE	GO	093H
	(MSB)							(LSB)	

Reset Status: DCON0 and DCON1 = 00H

Bit Definition:

DAS	IDA	Destination Address Space
0	0	External Data Memory without Auto-Increment
0	1	External Data Memory with Auto-Increment
1	0	Special Function Register
1	1	Internal Data Memory

SAS	ISA	Source Address Space
0	0	External Data Memory without Auto-Increment
0	1	External Data Memory with Auto-Increment
1	0	Special Function Register
1	1	Internal Data Memory

DM	TM	DMA Transfer Mode
0	0	Alternate-Cycle Transfer Mode
0	1	Burst Transfer Mode
1	0	FIFO or Serial Channel Demand Mode
1	1	External Demand Mode

DONE DMA transfer Flag:

0 DMA transfer is not completed.

1 DMA transfer is complete.

NOTE:
This flag is set when contents of the Byte Count SFR decrements to zero. It is reset automatically when the DMA vectors to its interrupt routine.

GO Enable DMA Transfer:

0 Disable DMA transfer (in all modes).

1 Enable DMA transfer. If the DMA is in the Block mode, start DMA transfer if possible. If it is in the Demand mode, enable the channel and wait for a demand.

NOTE:
The GO bit is reset when the BCR decrements to zero.

DMA Transfer Modes

The following four modes of DMA operation are possible in the UPI-452.

1. ALTERNATE-CYCLE MODE

General

Alternate cycle mode is useful when CPU processing must occur during the DMA transfers. In this mode, a DMA cycle and an instruction cycle occur alternately. The interrupt request is generated (if enabled) at the end of the process, i.e. when BCR decrements to zero. The transfer is initiated by setting the GO bit in the DCON SFR.

Alternate-Cycle FIFO Demand Mode

Alternate cycle demand mode is useful for FIFO transfers of a less urgent nature. As mentioned before, CPU instruction cycles are interleaved with DMA transfer cycles, allowing true parallel processing.

This mode differs from FIFO Demand Mode in that CPU instruction cycles must be interleaved with DMA transfers, even if the FIFO is demanding DMA. In FIFO Demand Mode, CPU cycles would never occur if the FIFO demand was present.

Input Channel

The DMA is configured as in FIFO Demand Mode and transfers are initiated whenever an Input FIFO service request is generated. DMA transfer cycles are alternated with instruction execution cycles. DMA transfers are terminated as in FIFO Demand Mode.

Output Channel

The DMA is configured as in FIFO Demand Mode and transfers are initiated whenever an Output FIFO requests service. DMA transfer cycles are alternated with instruction execution cycles. DMA transfers are terminated as in FIFO Demand Mode.

The FIFO logic resets the interrupt flag after transferring the byte, so the interrupt is never generated.

Once the DMA is programmed to service the FIFO, the request for service interrupt for the FIFO is inhibited until the DMA is done (BCR = 0).

2. BURST MODE

In BURST mode the DMA is initiated by setting the GO bit in the DCON SFR. The DMA operation continues until BCR decrements to zero (zero byte count), then an interrupt is generated (if enabled). No interrupts are recognized during a DMA operation once started.

Input Channel

The FIFO Input Channel can be used in burst mode by specifying the FIFO IN SFR as the DMA Source Address. DMA transfers begin when the GO bit in the DMA Control SFR is set. The number of bytes to be transferred must be specified in the Byte Count SFR (BCR) and auto-incrementing of the SAR must be disabled. Once the GO bit is set nothing can interrupt the transfer of data until the BCR is zero. In this mode, a Data Stream Command encountered in the FIFO will be held in the COMMAND IN SFR with the pointers frozen, and invalid data (FFH) will be read through the FIFO IN SFR. If the input FIFO becomes empty during the block transfer, an 0FFH will be read until BCR decrements to zero.

Output Channel

The Output FIFO Channel can be used in burst mode by specifying the FIFO OUT or COMMAND OUT SFR as the DMA Destination Address. DMA transfers begin when the GO bit is set. This mode can be used to send a block of data or a block of Data Stream Commands. If the FIFO becomes full during the block transfer, the remaining data will be lost.

NOTE:
All interrupts including FIFO interrupts are not recognized in Burst Mode. Burst Mode transfers should be used to service the FIFO only when the user is certain that no Data Stream Commands are in the block to be transferred (Input FIFO) and that the FIFO contains enough space to store the block to be transferred. In all other cases Alternate Cycle or Demand Mode should be used.

3. FIFO AND SERIAL CHANNEL DEMAND MODES

NOTES:
1. If the output FIFO is configured as a one byte buffer and the user program consists of two-cycle instructions only, then Alternate-Cycle Mode should be used.
2. In non-auto increment mode for internal to external, or external to internal transfers, the lower 8 bits of the external address should not correspond to the FIFO or Serial Port address.

FIFO Demand Mode

Although any DMA mode is possible using the FIFO buffer, only FIFO Demand and Alternate Cycle FIFO Demand Modes are recommended. FIFO Demand Mode DMA transfers using the input FIFO Channel are set-up by setting the GO bit and specifying the FIFO IN register as the DMA Source Address Register. The BCR should be set to the maximum number of expected transfers. The user must also program bit 1 of the Slave Control Register (SC1) to determine whether the Slave Status (SSTAT) SFR FIFO Request For Service Flag will be activated when the FIFO becomes not empty or full. Once the Request For Service Flag is activated by the FIFO, the DMA transfer begins, and continues until the request flag is deactivated. While the request is active, nothing can interrupt the DMA (i.e. it behaves like burst mode). The DMA Request is held active until one of the following occurs:

1) The FIFO becomes empty.
2) A Data Stream Command is encountered (this generates a FIFO interrupt and DMA operation resumes after the Data Stream Command is read).
3) BCR = 0 (this generates a DMA interrupt and sets the DONE bit).

DMA transfers to the Output FIFO Channel are similar. The FIFO OUT or COMMAND OUT SFR is the DMA Destination Address SFR and a transfer is started by setting the GO bit. The user programs bit 0 of the Slave Control SFR (SC0) to determine whether a demand occurs when the Output FIFO is not full or empty. DMA transfers begin when the Request For Service Flag is activated by the FIFO logic and continue as long as the flag is active. The Flag remains active until one of the following occurs:

1) The FIFO becomes full
2) BCR = 0 (this generates a DMA interrupt and sets the DONE bit).

As in Alternate Cycle FIFO Demand Mode, the FIFO logic resets the interrupt flag after transferring the byte, so the interrupt is never generated.

After the GO bit is set, the DMA is activated if one of the following conditions takes place:

SAR(0/1) = FIFO IN and HIFRS flag is set
DAR(0/1) = FIFO OUT and HOFRS flag is set

The HIFRS and HOFRS signals are internal flags which are not accessible by software. These flags are similar to the SST0 and SST4 flags in the Slave Status Register except that they are of the opposite polarity and once set they are not cleared until the Input FIFO becomes empty (HIFRS) or the Output FIFO becomes full (HOFRS).

Serial Channel Demand Mode

Serial Channel Demand Mode is the logical choice when using the Serial Port. The DMAs can be activated by one of the Serial Channel Flags. Receiver interrupt (RI) or Transmitter Interrupt (TI).

SAR(0/1) = SBUF and RI flag is set
DAR(0/1) = SBUF and TI flag is set

NOTE:
TI flag must be set by software to initiate the first transfer.

When the DMA transfer begins, only one byte is transferred at a time. The serial port hardware automatically resets the flag after completion of the transfer, so an interrupt will not be generated unless DMA servicing is held off due to the DMA being done (BCR = 0) or when the Hold/Hold Acknowledge logic is used and the DMA does not own the bus. In this case a Serial Port interrupt may be generated if enabled because of the status of the RI or TI flags.

In FIFO demand mode, Alternate cycle FIFO demand mode or Serial Port demand mode only one of the following registers (SBUF, FIN or FOUT) should be used as either the SAR or DAR registers to prevent undesired transfers. For example if SAR0 = FIN and DAR0 = SBUF in demand mode, the DMA transfer will start if either the HIFRS or TI flags are set.

4. EXTERNAL DEMAND MODE

The DMA can be initiated by an external device via External interrupt 0 and 1 (INT0/INT1) pins. The INT0 pin demands DMA0 (Channel 0) and INT1 demands DMA1 (Channel 1). If the interrupts are configured in edge mode, a single byte transfer is accomplished for every request. Interrupts also result (INT0 and INT1) after every byte transfer (if enabled). If the interrupts are configured in level mode, the DMA transfer continues until the request goes inactive or BCR = 0. In either case, a DMA interrupt is generated (if enabled) when BCR = 0. The GO bit must be set for the transfer to begin.

EXTERNAL MEMORY DMA

When transferring data to or from external memory via DMA, the HOLD (HLD) and HOLD-ACKNOWLEDGE (HLDA) signals are used for handshaking. The HOLD and HOLD-ACKNOWLEDGE are active low signals which arbitrate control of the local bus. The UPI-452 can be used in a system where multimasters are connected to a single parallel Address/Data bus. The HLD/HLDA signals are used to share resources (memory, peripherals, etc.) among all the processors on the local bus. The UPI-452 can be configured in any of three different External Memory Modes controlled by bits 5 and 6 (REQ & ARB) in the PCON SFR (Table 5). Each mode is described below:

REQUESTER MODE: In this mode, the UPI-452 is not the bus master, but must request the bus from another device. The UPI-452 configures port pin P1.5 as a HLD output and pin P1.6 as a HLDA input. The UPI-452 issues a HLD signal when it needs external access for a DMA channel. It uses the local bus after receiving the HLDA signal from the bus master, and will not release the bus until its DMA operation is complete.

ARBITER MODE: In this mode, the UPI-452 is the bus master. It configures port pin P1.5 as HLD input and pin P1.6 as HLDA output. When a device asserts the HLD signal to use the local bus, the UPI-452 asserts the HLDA signal after current instruction execution is complete. If the UPI-452 needs an external access via a DMA channel, it waits until the requester releases the bus, HLD goes inactive.

DISABLE MODE: When external program memory is accessed by an instruction or by program counter overflow beyond the internal ROM address or external data memory is accessed by MOVX instructions, it is a local memory access and the HLD/HLDA logic is not initiated. When a DMA channel attempts data transfer to/from the external data memory, the HLD/HLDA logic is initiated as described below. DMA transfers from the internal memory space to the internal memory space does not initiate the HLD/HLDA logic.

The balance of the PCON SFR bits are described in the "80C51 Register Description: Power Control SFR" section below.

Latency

When the GO bit is set, the UPI-452 finishes the current instruction before starting the DMA operation. Thus the maximum latency is 3.0 microseconds (at 16 MHz).

DMA Interrupt Vectors

Each DMA channel has a unique vectored interrupt associated with it. There are two vectored interrupts associated with the two DMA channels. The DMA interrupts are enabled and priorities set via the Interrupt Enable and Priority SFR (see "Interrupts" section). The interrupt priority scheme is similar to the scheme in 80C51.

Table 5. DMA MODE CONTROL - PCON SFR

Symbolic Address								Physical Address
PCON	—*	ARB	REQ	—*	—*	—*	—*	—*	87H
	(MSB)							(LSB)

*Defined as per MLS-51 Data Sheet
Reset Status: 00H

Definition:

ARB	REQ	
0	0	HLD/HLDA logic is disabled.
0	1	The UPI-452 is in the Requester Mode.
1	0	The UPI-452 is in the Arbiter Mode.
1	1	Invalid

When a DMA operation is complete (BCR decrements to zero), the DONE flag in the respective DCON (DCON0 or DCON1) SFR is set. If the DMA interrupt is enabled, the DONE flag is reset automatically upon vectoring to the interrupt routine.

Interrupts When DMA is Active

If a Burst Mode DMA transfer is in progress, the interrupts are not serviced until the DMA transfer is complete. This is also true for level activated External Demand DMA transfers. During Alternate Cycle DMA transfers, however, the interrupts are serviced at the end of the DMA cycle. After that, DMA cycles and instruction execution cycles occur alternately. In the case of edge activated External Demand Mode DMA transfers, the interrupt is serviced at the end of DMA transfer of that single byte.

DMA Arbitration

Only one of the two DMA channels is active at a time, except when both are configured in the Alternate Cycle mode. In this case, the DMA cycles and Instruction Execution cycles occur in the following order:

1. DMA Cycle 0.
2. Instruction execution.
3. DMA Cycle 1.
4. Instruction execution.

DMA0 has priority over DMA1 during simultaneous activation of the two DMA channels. If one DMA channel is active, the other DMA channel, if activated, waits until the first one is complete.

If DMA0 is already in the Alternate Cycle mode and DMA1 is activated in Alternate Cycle Mode, it will take two instruction cycles before DMA1 is activated (due to the priority of DMA0). Once DMA1 becomes active, the execution will follow the normal sequence.

If DMA0 is already in the Alternate Cycle mode and DMA1 is activated in Burst Mode, the DMA1 Burst transfer will follow the DMA0 Alternate Cycle transfer (after the completion of the next instruction).

If the UPI-452 (as a Requester) asserts a HLD signal to request a DMA transfer (see "External Memory DMA")and its other DMA Channel requests a transfer before the HLDA signal is received, the channel having higher priority is activated first. A Burst Mode transfer on channel 0 can not be interrupted since DMA0 has the highest priority. A Demand Mode transfer on channel 0 is the only type of activity that can interrupt a block transfer on DMA1.

If, while executing a DMA transfer, the Arbiter receives a HLD signal, and then before it can acknowledge, its other DMA Channel requests a transfer, it then completes the second DMA transfer before sending the HLDA signal to release the bus to the HLD request.

DMA transfers may be held off under the following conditions:

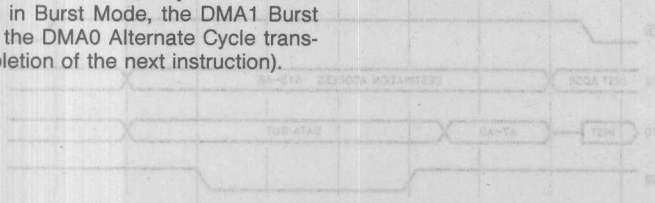
1. A write to any of the DMA registers inhibits the DMA for one instruction cycle.

NOTE:
An instruction cycle may be executed in 1, 2 or 4 machine cycles dependent on the instruction being executed. DMA transfers are only executed after the completion of an instruction cycle never between machine cycles of a single instruction cycle. Similarly instruction cycles are only executed upon completion of a DMA transfer whether it be a one machine cycle transfer or two machine cycles (for ext. to ext. memory transfers).

2. A single machine cycle DMA register read operation (i.e. MOV A, DCON0) will inhibit the DMA for one instruction cycle. However a two cycle DMA register read operation will not inhibit the DMA (i.e. MOV P1, DCON0).

If the HOLD/HOLD Acknowledge logic is enabled in requestor mode the hold request will go active once the go bit has been set (for burst mode) and once the demand flag is set (for demand mode) regardless of whether the DMA is held off by one of the above conditions.

The DMA Transfer waveforms are in Figures 8-11.



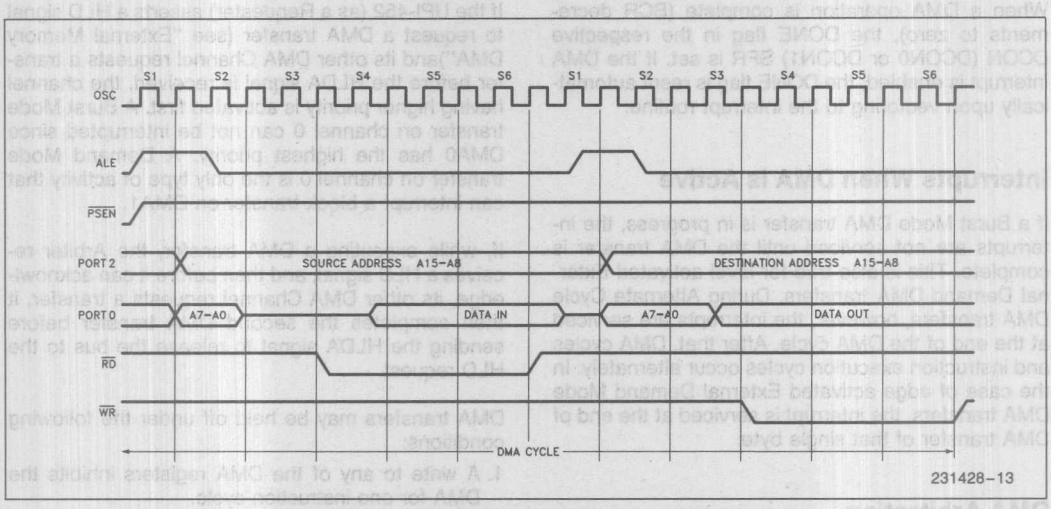


Figure 8. DMA Transfer from External Memory to External Memory

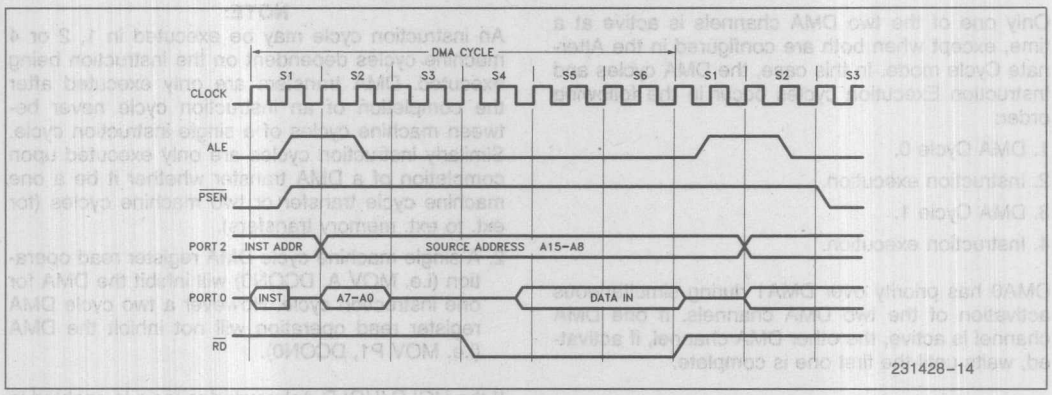


Figure 9. DMA Transfer from External Memory to Internal Memory

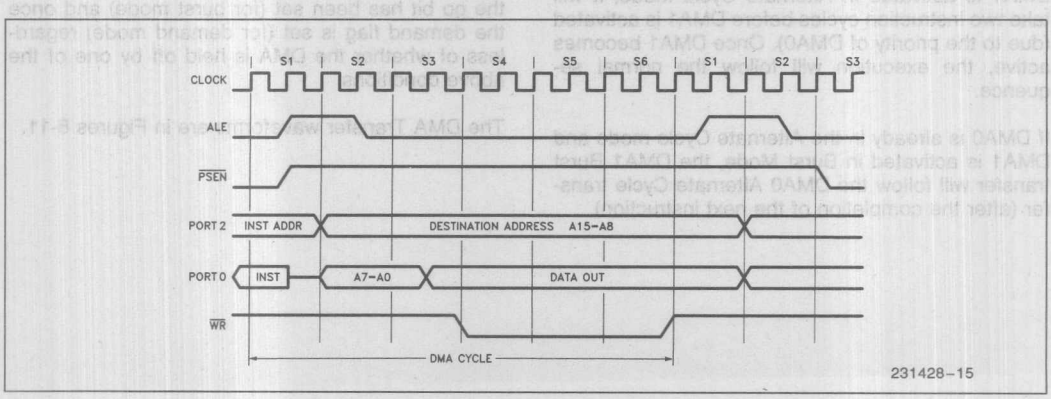


Figure 10. DMA Transfer from Internal Memory to External Memory

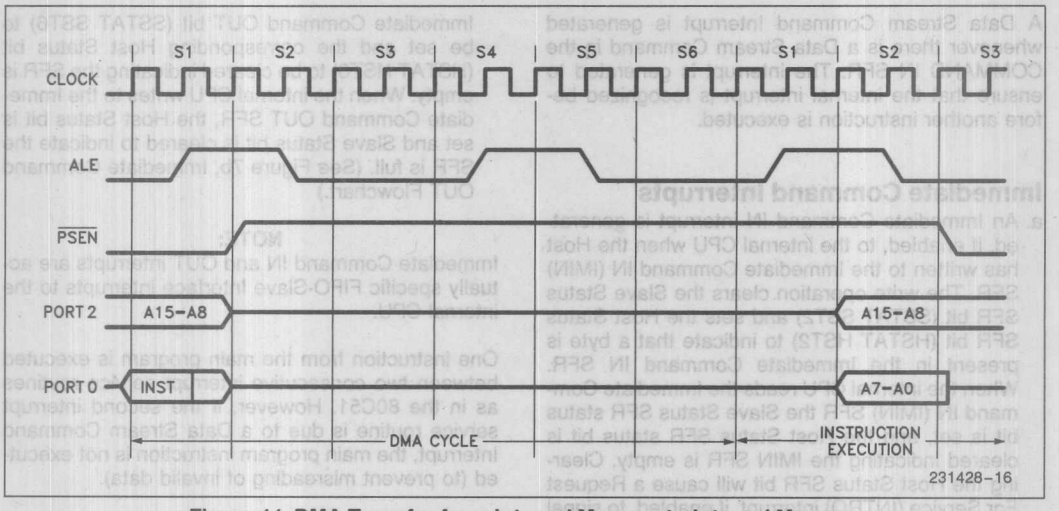


Figure 11. DMA Transfer from Internal Memory to Internal Memory

INTERNAL INTERRUPTS

Overview

The UPI-452 provides a total of eight interrupt sources (Table 6). Their operation is the same as in the 80C51, with the addition of three new interrupt sources for the UPI-452 FIFO and DMA features. These added interrupts have their enable and priority bits in the Interrupt Enable and Priority (IEP) SFR. The IEP SFR is in addition to the 80C51 Interrupt Enable (IE) and Interrupt Priority (IP) SFRs. The added interrupt sources are also globally enabled or disabled by the EA bit in the Interrupt Enable SFR. Table 6 lists the eight interrupt sources in order of priority. Table 7 lists the eight interrupt sources and their respective address vector location in program memory. (DMA interrupts are discussed in the "General Purpose DMA Channels" section. Additional interrupt information for Timer/Counter, Serial Channel, External Interrupt may be found in the Microcontroller Handbook for the 80C51.)

Table 6. Interrupt Priority

Interrupt Source	Priority Level
	(highest)
External Interrupt 0	0
Internal Timer/Counter 0	1
DMA Channel 0 Request	2
External Interrupt 1	3
DMA Channel 1 Request	4
Internal Timer/Counter 1	5
FIFO - Slave Bus Interface	6
Serial Channel	7
	(lowest)

Table 7. Interrupt Vector Addresses

Interrupt Source	Starting Address
External Interrupt 0	3 (003H)
Internal Timer/Counter 0	11 (00BH)
External Interrupt 1	19 (013H)
Internal Timer/Counter 1	27 (01BH)
Serial Channel	35 (023H)
FIFO - Slave Bus Interface	43 (02BH)
DMA Channel 0 Request	51 (033H)
DMA Channel 1 Request	59 (03BH)

FIFO Module Interrupts to Internal CPU

The FIFO module generates interrupts to the internal CPU whenever the FIFO requests service or when a Data Stream Command is in the COMMAND IN SFR. The Input FIFO will request service whenever it becomes full or not empty depending on bit 1 of the Slave Control SFR (IFRS). Similarly, the Output FIFO requests service when it becomes empty or not full as determined by bit 0 of the Slave Control SFR (OFRS). Request for Service interrupts are generated only if enabled by the internal CPU via the Interrupt Enable SFR, and the Slave Control Register.

whenever there is a Data Stream Command in the COMMAND IN SFR. The interrupt is generated to ensure that the internal interrupt is recognized before another instruction is executed.

Immediate Command Interrupts

a. An Immediate Command IN interrupt is generated, if enabled, to the internal CPU when the Host has written to the Immediate Command IN (IMIN) SFR. The write operation clears the Slave Status SFR bit (SSTAT SST2) and sets the Host Status SFR bit (HSTAT HST2) to indicate that a byte is present in the Immediate Command IN SFR. When the internal CPU reads the Immediate Command IN (IMIN) SFR the Slave Status SFR status bit is set, and the Host Status SFR status bit is cleared indicating the IMIN SFR is empty. Clearing the Host Status SFR bit will cause a Request For Service (INTRQ) interrupt, if enabled, to signal the Host that the IMIN SFR is empty. (See Figure 7a, Immediate Command IN Flowchart.)

b. An Immediate Command OUT interrupt is generated, if enabled, to the internal CPU when the Host has read the Immediate Command OUT SFR. The Host read causes the Slave Status Immediate Command OUT bit (SSTAT SST6) to be set and the corresponding Host Status bit (HSTAT HST6) to be cleared indicating the SFR is empty. When the internal CPU writes to the Immediate Command OUT SFR, the Host Status bit is set and Slave Status bit is cleared to indicate the SFR is full. (See Figure 7b, Immediate Command OUT Flowchart.)

NOTE:
Immediate Command IN and OUT interrupts are actually specific FIFO-Slave Interface interrupts to the internal CPU.

One instruction from the main program is executed between two consecutive interrupt service routines as in the 80C51. However, if the second interrupt service routine is due to a Data Stream Command Interrupt, the main program instruction is not executed (to prevent misreading of invalid data).

Interrupt Enabling and Priority

Each of the three interrupt special function registers (IE, IP and IEP) is listed below with its corresponding bit definitions.

Interrupt Enable SFR (IE)

Symbolic Address									Physical Address
IE	EA	—	—	ES	ET1	EX1	ET0	EX0	0A8H
	(MSB)							(LSB)	

Symbol	Position	Function
EA	IE.7	Enables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	(reserved)
—	IE.5	(reserved)
ES	IE.4	Serial Channel interrupt enable
ET1	IE.3	Internal Timer/Counter 1 Overflow Interrupt
EX1	IE.2	External Interrupt Request 1.
ET0	IE.1	Internal Timer/Counter 0 Overflow Interrupt
EX0	IE.0	External Interrupt Request 0.

Interrupt Priority SFR (IP)

A priority level of 0 or 1 may be assigned to each interrupt source, with 1 being higher priority level, through the IP and the IEP (Interrupt Enable and Priority) SFR. A priority level of 1 interrupt can interrupt a priority level 0 service routine to allow nesting of interrupts.

Symbolic Address									Physical Address
IP	—	—	—	PS	PT1	PX1	PT0	PX0	0B8H
	(MSB)							(LSB)	

Symbol	Position	Function	Priority Within A Level
			(lowest)
—	IP.7	(reserved)	—
—	IP.6	(reserved)	—
—	IP.5	(reserved)	—
PS	IP.4	Local Serial Channel	0.7
PT1	IP.3	Internal Timer/Counter 1	0.5
PX1	IP.2	External Interrupt Request 1	0.3
PT0	IP.1	Internal Timer/Counter 0	0.1
PX0	IP.0	External Interrupt Request 0	0.0
			(highest)

Interrupt Enable and Priority SFR (IEP)

The Interrupt Enable and Priority Register establishes the enabling and priority of those resources not covered in the Interrupt Enable and Interrupt Priority SFRs.

Symbolic Address									Physical Address
IEP	—	—	PFIFO	EDMA0	EDMA1	PDMA0	PDMA1	EFIFO	0F8H
	(MSB)							(LSB)	

Symbol	Position	Function	Priority Within a Level
—	IEP.7	(reserved)	
—	IEP.6	(reserved)	
PFIFO	IEP.5	FIFO Slave Bus Interface Interrupt Priority	0.6
EDMA0	IEP.4	DMA Channel 0 Interrupt Enable	
EDMA1	IEP.3	DMA Channel 1 Interrupt Enable	
PDMA0	IEP.2	DMA Channel 0 Priority	0.2
PDMA1	IEP.1	DMA Channel 1 Priority	0.4
EFIFO	IEP.0	FIFO Slave Bus Interface Interrupt Enable	

FIFO-EXTERNAL HOST INTERFACE FIFO DMA FREEZE MODE

Overview

During FIFO DMA Freeze Mode the internal CPU can reconfigure the FIFO interface. FIFO DMA Freeze Mode is provided to prevent the Host from accessing the FIFO during a reconfiguration sequence. The internal CPU invokes FIFO DMA Freeze Mode by clearing bit 3 of the Slave Control SFR (SC3). INTRQ becomes active whenever FIFO DMA Freeze Mode is invoked to indicate the freeze status. The interrupt can only be deactivated by the Host reading the Host Status SFR.

During FIFO DMA Freeze Mode only two operations are possible by the Host to the UPI-452 slave, the balance are disabled, as shown in Table 8. The internal DMA is disabled during FIFO DMA Freeze Mode, and the internal CPU has write access to all of the FIFO control SFRs (Table 9).

Initialization

At power on reset the FIFO Host interface is automatically frozen. The Slave Control Enable FIFO DMA Freeze Mode bit defaults to FIFO DMA Freeze Mode (SLCON FRZ = 0). Below is a list of the FIFO Special Function Registers and their default power on reset values;

SFR Name	Label	Value
Channel Boundary Pointer	CBP	40H / 64D
Output Channel Read Pointers	ORPR	40H / 64D
Output Channel Write Pointers	OWPR	40H / 64D
Input Channel Read Pointers	IRPR	00H / 00D
Input Channel Write Pointers	IWPR	00H / 00D
Input Threshold	ITHR	80H / 128D
Output Threshold	OTHR	01H / 1D

The Input and Output FIFO channels may be reconfigured by programming any of these SFRs while the FIFO Host interface is in FIFO DMA Freeze Mode. The UPI-452 also notifies the Host that FIFO DMA Freeze Mode is in progress by setting the Host Status SFR FIFO DMA Freeze Mode Status bit, FIFO DMA Freeze Mode In Progress. The Host interrogates the Host Status SFR to determine the status of the FIFO Host interface following reset before attempting to read from or write to the UPI-452 FIFO buffer.

Table 8. Slave Bus Interface Status During FIFO DMA Freeze Mode

Interface Pins; $\overline{DACK}$	$\overline{CS}$	A2	A1	A0	$\overline{READ}$	$\overline{WRITE}$	Operation In Normal Mode	Status In FIFO DMA Freeze Mode
1	0	0	1	0	0	1	Read Host Status SFR	Operational
1	0	0	1	1	0	1	Read Host Control SFR	Operational
1	0	0	1	1	1	0	Write Host Control SFR	Disabled
1	0	0	0	0	0	1	Data or DMA Data from Output Channel	Disabled
1	0	0	0	0	1	0	Data or DMA Data to Input Channel	Disabled
1	0	0	0	1	0	1	Data Stream Command from Output Channel	Disabled
1	0	0	0	1	1	0	Data Stream Command to Input Channel	Disabled
1	0	1	0	0	0	1	Read Immediate Command Out from Output Channel	Disabled
1	0	1	0	0	1	0	Write Immediate Command In to Input Channel	Disabled
0	X	X	X	X	0	1	DMA Data from Output Channel	Disabled
0	X	X	X	X	1	0	DMA Data to Input Channel	Disabled

The UPI-452 can also be programmed to interrupt the Host following power on reset in order to indicate to the Host that FIFO DMA Freeze Mode is in progress. This is done by enabling the INTRQ interrupt output pin via the MODE SFR (MD4) before the Slave Control SFR Enable FIFO DMA Freeze Mode bit is set to Normal Mode. At power on reset the Mode SFR is forced to zero. This disables all interrupt and DMA output pins (INTRQ, DRQIN/INTRQIN and DRQOUT/INTRQOUT). Because the Host Status SFR FIFO DMA Freeze Mode In Progress bit is set, a Request For Service, INTRQ, interrupt is pending until the Host Status SFR is read. This is because the FIFO DMA Freeze Mode interrupt is always enabled. If the Slave Control FIFO DMA Freeze Mode bit (SLCON FRZ) is set to Normal Mode before the MODE SFR INTRQ bit is enabled, the INTRQ output will not go active when the MODE SFR INTRQ bit is enabled if the Host Status SFR has been read.

The default values for the FIFO and Slave Interface represents minimum UPI-452 internal initialization. No specific Special Function Register initialization is required to begin operation of the FIFO Slave Interface. The last initialization instruction must always set the UPI-452 to Normal Mode. This causes the UPI-452 to exit FIFO DMA Freeze Mode and enables Host read/write access of the FIFO.

Following reset, either hardware (via the RST pin) or software (via HCON SFR bit HC3) the UPI-452 requires 2 internal machine cycles (24 TCLCL) to update all internal registers.

Invoking FIFO DMA Freeze Mode During Normal Operation

When the UPI-452 is in normal operation, FIFO DMA Freeze Mode should not be arbitrarily invoked by clearing SC3 (SC3 = 0) because the external Host runs asynchronously to the internal CPU. Invoking FIFO DMA Freeze Mode without first stopping the external Host from accessing the UPI-452 will not guarantee a clean break with the external Host.

The proper way to invoke FIFO DMA Freeze Mode is by issuing an Immediate Command to the external host indicating that FIFO DMA Freeze Mode will be invoked. Upon receiving the Immediate Command, the external Host should complete servicing all pending interrupts and DMA requests, then send an Immediate Command back to the UPI-452 acknowledging the FIFO DMA Freeze Mode request. After issuing the first Immediate Command, the internal CPU should not perform any action on the FIFO until FIFO DMA Freeze Mode is invoked.

If FIFO DMA Freeze Mode is invoked without stopping the Host during Host transfers, only the last two bytes of data written into or read from the FIFO will be valid. The timing diagram for disabling the FIFO module to the external Host interface is illustrated in Figure 12. Due to this synchronization sequence, the UPI-452 might not go into FIFO DMA Freeze Mode immediately after SC3 is cleared. A special bit in the Slave Status Register (SST5) is provided to indicate the status of the FIFO DMA Freeze Mode. The FIFO DMA Freeze Mode operations described in this section are only valid after SST5 is cleared.

As FIFO DMA Freeze Mode is invoked, the DRQIN or DRQOUT will be deactivated (stopping the transferring of data), bit 1 of the Host Status SFR will be set (HST1 = 1), and SST5 will be cleared (SST5 = 0) to indicate to the external Host and internal CPU that the slave interface has been frozen. After the freeze becomes effective, any attempt by the external Host to access the FIFO will cause the overrun and underrun bits to be activated (bits HST7 (for reads) or HST3 (for writes)). These two bits, HST3 and HST7, will be set (deactivated) after the Host Status SFR has been read. If INTRQ is used to request service, the FIFO interface is frozen upon completion of any Host read or write operation in progress.

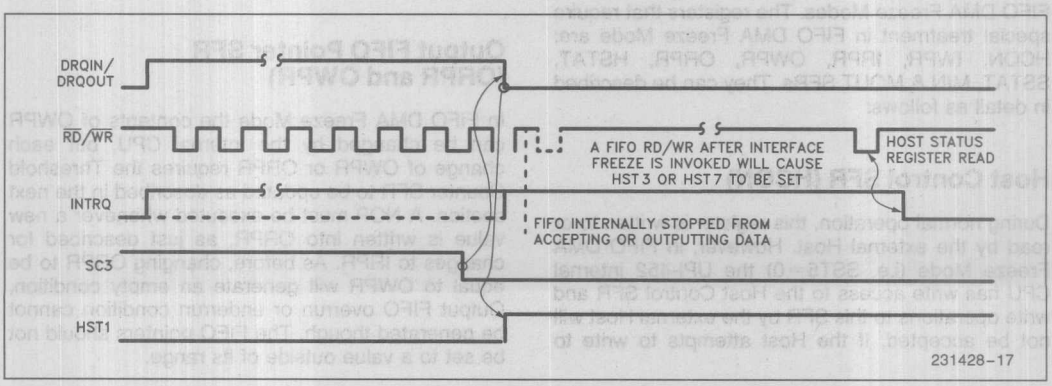


Figure 12. Disabling FIFO to Host Slave Interface Timing Diagram

External Host writing to the Immediate Command In SFR and the Host Control SFR is also inhibited when the slave bus interface is frozen. Writing to these two registers after FIFO DMA Freeze Mode is invoked will also cause HST3 (overrun) to be activated (HST3 = 0). Similarly, reading the Immediate Command Out Register by the external Host is disabled during FIFO DMA Freeze Mode, and any attempt to do so will cause the clearing (deactivating, "0") of HST7 bit (underrun).

After the slave bus interface is frozen, the internal CPU can perform the following operations on the FIFO Special Function Registers (these operations are allowed only during FIFO DMA Freeze Mode).

For FIFO Reconfiguration	1. Changing the Channel Boundary Pointer SFR. 2. Changing the Input and Output Threshold SFR.
To Enhance the Testability	3. Writing to the read and write pointers of the Input and Output FIFO's. 4. Writing to and reading the Host Control SFRs. 5. Controlling some bits of Host and Slave Status SFRS. 6. Reading the Immediate Command Out SFR and Writing to the Immediate Comand In SFR.

Description of each of these special functions are as follows:

FIFO Module SFRs During FIFO DMA Freeze Mode

Table 9 summarizes the characteristics of all the FIFO Special Function Registers during normal and FIFO DMA Freeze Modes. The registers that require special treatment in FIFO DMA Freeze Mode are: HCON, IWPR, IRPR, OWPR, ORPR, HSTAT, SSTAT, MIN & MOUT SFRs. They can be described in detail as follows:

Host Control SFR (HCON)

During normal operation, this register is written to or read by the external Host. However, in FIFO DMA Freeze Mode (i.e. SST5 = 0) the UPI-452 internal CPU has write access to the Host Control SFR and write operations to this SFR by the external Host will not be accepted. If the Host attempts to write to HCON, the Input Channel error condition flag (HST3) will be cleared.

Input FIFO Pointer Registers (IRPR & IWPR)

Once the FIFO module is in FIFO DMA Freeze Mode, error flags due to overrun and underrun of the Input FIFO pointers will be disabled. Any attempt to create an overrun or underrun condition by changing the Input FIFO pointers would result in an inconsistency in performance between the status flag and the threshold counter.

To enhance the speed of the UPI-452, read operations on the Input FIFO will look ahead by two bytes. Hence, every time the IRPR is changed during FIFO DMA Freeze Mode, two NOPs need to be executed so that the two byte pipeline can be updated with the new data bytes pointed to by the new IRPR. The Threshold Counter SFR also needs to change by the same number of bytes as the IRPR (increase Threshold Counter if IRPR goes forward or decrease if IRPR goes backward). This will ensure that future interrupts will still be generated only after a threshold number of bytes are available. (See "Input and Output FIFO Threshold SFR" section below.)

In FIFO DMA Freeze Mode, the internal CPU can also change the content of IWPR, and each change of IWPR also requires an update of the Threshold Counter SFR.

Normally, the internal CPU cannot write into the Input FIFO. It can, however, during FIFO DMA Freeze Mode by first reconfiguring the FIFO as an Output FIFO (Refer to "Input and Output FIFO Threshold SFR" section below). Changing the IRPR to be equal to IWPR generates an empty condition while changing IWPR to be equal to IRPR generates a full condition. The order in which the pointers are written determines whether a full or empty condition is generated.

Output FIFO Pointer SFR (ORPR and OWPR)

In FIFO DMA Freeze Mode the contents of OWPR can be changed by the internal CPU, but each change of OWPR or ORPR requires the Threshold Counter SFR to be updated as described in the next section. A NOP must be executed whenever a new value is written into ORPR, as just described for changes to IRPR. As before, changing ORPR to be equal to OWPR will generate an empty condition, Output FIFO overrun or underrun condition cannot be generated though. The FIFO pointers should not be set to a value outside of its range.

Table 9. FIFO SFR's Characteristics During FIFO DMA Freeze Mode

Label	Name	Normal Operation (SST5 = 1)	FIFO DMA Freeze Mode Operation (SST5 = 0)
HCON	Host Control	Not Accessible	Read & Write
HSTAT	Host Status	Read Only	Read & Write 4
SLCON	Slave Control	Read & Write	Read & Write
SSTAT	Slave Status	Read Only	Read & Write 4
IEP	Interrupt Enable & Priority	Read & Write	Read & Write
MODE	Mode Register	Read & Write	Read & Write
IWPR	Input FIFO Write Pointer	Read Only	Read & Write 5
IRPR	Input FIFO Read Pointer	Read Only	Read & Write 1, 5
OWPR	Output FIFO Write Pointer	Read Only	Read & Write 6
ORPR	Output FIFO Read Pointer	Read Only	Read & Write 2, 6
CBP	Channel Boundary Pointer	Read Only	Read & Write 3
IMIN	Immediate Command In	Read Only	Read & Write
IMOUT	Immediate Command Out	Read & Write	Read & Write
FIN	FIFO IN	Read Only	Read Only
CIN	COMMAND IN	Read Only	Read Only
FOUT	FIFO OUT	Read & Write	Read & Write
COUT	COMMAND OUT	Read & Write	Read & Write
ITHR	Input FIFO Threshold	Read Only	Read & Write
OTHR	Output FIFO Threshold	Read Only	Read & Write

NOTES:
1. Writing of IRPR will automatically cause the FIFO IN SFR to load the contents of the Input FIFO from that location.
2. Writing to ORPR will automatically cause the IOBL SFR to load the contents of the Output FIFO at that ORPR address.
3. Writing to the CBP SFR will cause automatic reset of the four pointers of the Input and Output FIFO channels.
4. The internal CPU cannot directly change the status of these registers. However, by changing the status of the FIFO channels, the internal CPU can indirectly change the contents of the status registers.
5. Changing the Input FIFO Read/Write Pointers also requires that a consistent update of the Input FIFO Threshold Counter SFR.
6. Changing the Output FIFO Read/Write Pointers also requires that a consistent update of the Output FIFO Threshold Counter SFR.

Input and Output FIFO Threshold SFR (ITHR & OTHR)

The Input and Output FIFO Threshold SFRs are also programmable by the internal CPU during FIFO DMA Freeze Mode. For proper operation of the Threshold feature, the Threshold SFR should be changed only when the Input and Output FIFO channels are empty, since they reflect the current number of bytes available to read/write before an interrupt is generated.

Table 10 illustrates the Threshold SFRs range of values and the number of bytes to be transferred when the Request For Service Flag is activated:

Table 10. Threshold SFRs Range of Values and Number of Bytes to be Transferred

ITHR (lower seven bits)	No. of Bytes Available to be Written	OTHR (lower seven bits	No. of Bytes Available to be Read
0	CBP	1	2
1	CBP-1	2	3
2	CBP-2	3	4
•	•	•	•
•	•	•	•
•	•	•	•
CBP-3	3	(80H-CBP)-3	(80H-CBP)-2
CBP-2	2	(80H-CBP)-2	(80H-CBP)-1
		(80H-CBP)-1	(80H-CBP)

The eighth bit of the Input and Output FIFO Threshold SFR indicates the status of the service requests regardless of the freeze condition. If the eighth bit is a "1", the FIFO is requesting service from the external Host. In other words, when the Threshold SFR value goes below zero (2's complement), a service request is generated*. Normally the ITHR SFR is decremented after each external Host write to the Input FIFO and incremented after each internal CPU read of the Input FIFO. The OTHR SFR is decremented by internal CPU writes and incremented by external Host reads. Thus if the pointers are moved when the FIFO's are not empty, these relationships can be used to calculate the offset for the Threshold SFRs. It is best to change the Threshold SFRs only when the FIFO's are empty to avoid this complication. The threshold registers should also be updated after the pointers have been manipulated.

NOTE:
When programming the ITHR SFR, the eighth bit should be set to 1 (OR'd with 80H). This causes HSTAT SFR HST0 = 0, Input FIFO Request For Service. If ITHR bit 7 = 0 then HSTAT HST0 = 1, Input FIFO Does Not Request Service, and no interrupt will be generated.

*The 8th bit of the ITHR SFR must be set during initialization if the Host interrupt request is desired immediately upon leaving Freeze Mode.

Host Status SFR (HSTAT)

When in FIFO DMA Freeze Mode, some bits in the Host Status SFR are forced high and will not reflect the new status until the system returns to normal operation. The definition of the register in FIFO DMA Freeze Mode is as follows:

NOTE:
The internal CPU reads this shadow latch value when reading the Host Status SFR. The shadow latch will keep the information for these bits so normal operation can be resumed with the right status. The following bits are set (= 1) when FIFO DMA Freeze Mode is invoked;

HST7 Output FIFO Error Condition Flag

1 = No error.

0 = An invalid read has been done on the output FIFO or the Immediate Command Out Register by the host CPU.

NOTE:
The normal underrun error condition status is disabled. If an Immediate Command Out (IMOUT) SFR read is attempted during FIFO DMA Freeze Mode, the contents of the IMOUT SFR is output on the Data Buffer and the error status is cleared (= 0).

HST6 Immediate Command Out SFR Status

During normal operation, this bit is cleared (=0) when the IMOUT SFR is written by the UPI-452 internal CPU and set (= 1) when the IMOUT SFR is read by the external Host. Once the host-slave interface is frozen (i.e. SST5 = 0), this bit will be read as a 1 by the host CPU. A shadow latch will keep the information for this bit so normal operation can be resumed with the correct status.

Shadow latch:

1 = Internal CPU reads the IMOUT SFR

0 = Internal CPU writes to the IMOUT SFR

HST5 Data Stream Command at Output FIFO

This bit is forced to a "1" during FIFO DMA Freeze Mode to prevent the external host CPU from trying to read the DSC. Once normal operation is resumed, HST5 will reflect the Data/Command status of the current byte in the Output FIFO.

Shadow Latch (read by the internal CPU):

1 = No Data Stream Command (DSC)

0 = Data Stream Command at Output FIFO

HST4 Output FIFO Service Request Status

When FIFO DMA Freeze Mode is invoked, this bit no longer reflects the Output FIFO Request Service Status. This bit wll be forced to a "1".

HST3 Input FIFO Error Condition Flag

1 = No error.

0 = One of the following operations has been attempted by the external host and is invalid:

1) Write into the Input FIFO
2) Write into the Host Control SFR
3) Write into the Immediate Command In SFR

NOTE:
The normal Input FIFO overrun condition is disabled.

HST2 Immediate Command In SFR Status

This bit is normally cleared when the internal CPU reads the IMIN SFR and set when the external host CPU writes into the IMIN SFR. When the host-slave interface is frozen, reading and writing of the IMIN by the internal CPU will change the shadow latch of this bit. This bit will be read as a "1" by the external Host.

Shadow latch.

1 = Internal CPU writes into IMIN SFR

0 = Internal CPU reads the IMIN SFR

HST1 FIFO DMA Freeze Mode Status

1 = FIFO DMA Freeze Mode.

0 = Normal Operation (non-FIFO DMA Freeze Mode).

NOTE:
This bit is used to indicate to the external Host that the host-slave interface has been frozen and hence the external Host functions are now reduced as shown in Table 8.

HST0 Input FIFO Request Service Satus

When slave interface is frozen this bit no longer reflects the Input FIFO Request Service Status. This bit will be forced to a "1".

Slave Status SFR (SSTAT)

The Slave Status SFR is a read-only SFR. However, once the slave interface is frozen, most of the bits of this SFR can be changed by the internal CPU by reconfiguring the FIFO and accessing the FIFO Special Function Registers.

SST7 Output FIFO Overrun Error Flag

Inoperative in FIFO DMA Freeze Mode.

SST6 Immediate Command Out SFR Status

In FIFO DMA Freeze Mode, this bit will be cleared when the internal CPU reads the Immediate Command Out SFR and set when the internal CPU writes to the Immediate Command Out Register.

SST5 FIFO-External Interface FIFO DMA Freeze Mode Status

This bit indicates to the internal CPU that FIFO DMA Freeze Mode is in progress and that it has write access to the FIFO Control, Host control and Immediate Command SFRs.

SST4 Output FIFO Request Service Status

During normal operation, this bit indicates to the internal CPU that the Output FIFO is ready for more data. The status of this bit reflects the position of the Output FIFO read and write pointers. Hence, in FIFO DMA Freeze Mode, this flag can be changed by the internal CPU indirectly as the read and write pointers change.

SST3 Input FIFO Underrun Flag

Inoperative during FIFO DMA Freeze Mode.

During normal operation, a read operation clears (=0) this bit when there are no data bytes in the Input FIFO and deactivated (=1) when the Slave Status SFR is read. In FIFO DMA Freeze Mode, this bit will not be cleared by an Input FIFO read underrun error condition, nor will it be reset by the reading of the Slave Status SFR.

SST2 Immediate Command In SFR Status

This bit is normally activated (=0) when the external host CPU writes into the Immediate Command In SFR and deactivated (=1) when it is read by the internal CPU. In FIFO DMA Freeze Mode, this bit will not be activated (=0) by the external Host's writing of the Immediate Command IN SFR since this function is disabled. However, this bit will be cleared (=0) if the internal CPU writes to the Immediate Command In SFR and it will be set =1) if it reads from the register.

SST1 Data Stream Command at Input FIFO Flag

In FIFO DMA Freeze Mode, this bit operates normally. It indicates whether the next byte of data from the Input FIFO is a DSC or data byte. If it is a DSC byte, reading from the FIFO IN SFR will result in reading invalid data (FFH) and vice versa. In FIFO DMA Freeze Mode, this bit still reflects the type of data byte available from the Input FIFO.

SST0 Input FIFO Service Request Flag

During normal operation, this bit is activated (=0) when the Input FIFO contains bytes that can be read by the internal CPU and deactivated (=1) when the Input FIFO does not need any service from the internal CPU. In FIFO DMA Freeze Mode, the status of this bit should not change unless the pointers of the Input FIFO are changed. In this mode, the internal CPU can indirectly change this bit by changing the read and write pointers of the Input FIFO but cannot change it directly.

Immediate Command In/Out SFR (IMIN/IMOUT)

If FIFO DMA Freeze Mode is in progress, writing to the Immediate Command In SFR by the external host will be disabled, and any such attempt will cause HST3 to be cleared (=0). Similarly, the Immediate Command Out SFR read operation (by the host) will be disabled internally and read attempts will cause HST7 to be cleared (=0).

Internal CPU Read and Write of the FIFO During FIFO DMA Freeze Mode

In normal operation, the Input FIFO can only be read by the internal CPU and similarly, the Output FIFO can only be written by the internal CPU. During FIFO DMA Freeze Mode, the internal CPU can read the entire contents of the Input FIFO by programming the CBP SFR to 7FH, setting the IRPR SFR to zero, and then the IWPR SFR to zero. Programming the pointer registers in this order generates a FIFO full signal to the FIFO logic and enables internal CPU read operations. If the IWPR and IRPR are already zero, the write pointer should be changed to a non-zero value to clear the empty status then the pointers can be set to zero. Writing to the IRDR SFR automatically updates the look ahead registers.

In a similar manner, the internal CPU can write to all 128 bytes of the FIFO by setting the CBP SFR to zero, setting OWPR SFR to zero, and then setting ORPR SFR to zero. This generates a FIFO empty signal and allows internal CPU write operations to all 128 bytes of the FIFO. The Threshold registers also need to be adjusted when the pointers are changed. (See "Input and Output FIFO Threshold SFR" section below.)

MEMORY ORGANIZATION

The UPI-452 has separate address spaces for Program Memory and Data Memory like the 80C51. The Program Memory can be up to 64K bytes. The lower 8K of Program Memory may reside on-chip. The Data Memory consists of 256 bytes of on-chip RAM, up to 64K bytes of off-chip RAM and a number of "SFRs" (Special Function Registers) which appear as yet another set of unique memory addresses.

Table 11a. Internal Memory Addressing

Memory Space	Addressing Method
Lower 128 Bytes of Internal RAM	Direct or Indirect
Upper 128 Bytes of Internal RAM	Indirect Only
UPI-452 SFR's	Direct Only

The 80C51 Special Function Registers are listed in Table 11a, and the additional UPI-452 SFRs are listed in Table 11b. A brief description of the 80C51 core SFRs is also provided below.

Accessing External Memory

As in the 80C51, accesses to external memory are of two types: Accesses to external Program Memory and accesses to external Data Memory.

External Program Memory is accessed under two conditions:

1) Whenever signal $\overline{EA}$ = 0; or
2) Whenever the program counter (PC) contains a number that is larger than 1FFFH.

This requires that the ROMless versions have $\overline{EA}$ wired low to enable the lower 8K program bytes to be fetched from external memory.

External Data Memory is accessed using either the MOVX @DPTR (16 bit address) or the MOVX @Ri (8 bit address) instructions, or during external data memory transfers.

Table 11b. 80C51 Special Function Registers

Symbol	Name	Address	Contents
*ACC	Accumulator	0E0H	00H
*B	B Register	0F0H	00H
*PSW	Program Status Word	0D0H	00H
SP	Stack Pointer	81H	07H
DPTR	Data Pointer (consisting of DPH and DPL)	82H	0000H
*P0	Port 0	80H	0FFH
*P1	Port 1	90H	0FFH
*P2	Port 2	0A0H	0FFH
*P3	Port 3	0B0H	0FFH
*IP	Interrupt Priority Control	0B8H	0E0H
*IE	Interrupt Enable Control	0A8H	60H
TMOD	Timer/Counter Mode Control	89H	00H
*TCON	Timer/Counter Control	88H	00H
TH0	Timer/Counter 0 (high byte)	8CH	00H
TL0	Timer/Counter 0 (low byte)	8AH	00H
TH1	Timer/Counter 1 (high byte)	8DH	00H
TL1	Timer/Counter 1 (low byte)	8BH	00H
*SCON	Serial Control	98H	00H
SBUF	Serial Data Buff	99H	I
PCON	Power Control	87H	I0H

I = Indeterminate

Table 11c. UPI-452 Additional Special Function Registers

Symbol	Name	Address	Contents
BCRL0	DMA Byte Count Low Byte/	0E2H	I
BCRH0	High Byte/ Channel 0	0E3H	I
BCRL1	Low Byte/	0F2H	I
BCRH1	Hi Byte/ Channel 1	0F3H	I
CBP	Channel Boundary Pointer	0ECH	40H
CIN	COMMAND IN	0EFH	I
COUT	COMMAND OUT DMA Destination Address	0FFH	I

Table 11c. UPI-452 Additional Special Function Registers (Continued)

Symbol	Name	Address	Contents
DARL0	Low Byte/	0C2H	I
DARH0	Hi Byte/ Channel 0	0C3H	I
DARL1	Low Byte/	0D2H	I
DARH1	Hi Byte/ Channel 1	0D3H	I
DCON0	DMA0 Control	92H	00H
DCON1	DMA1 Control	93H	00H
FIN	FIFO IN	0EEH	I
FOUT	FIFO OUT	0FEH	I
HCON	Host Control	0E7H	00H
HSTAT	Host Status	0E6H	0FBH
*IEP	Interrupt Enable and Priority	0F8H	0C0H
IMIN	Immediate Command In	0FCH	I
IMOUT	Immediate Command Out	0FDH	I
IRPR	Input Read Pointer	0EBH	00H
ITHR	Input FIFO Threshold	0F6H	80H
IWPR	Input Write Pointer	0EAH	00H
MODE	Mode Register	0F9H	8FH
ORPR	Output Read Pointer	0FAH	40H
OTHR	Output FIFO Threshold	0F7H	01H
OWPR	Output Write Threshold	0FBH	40H
*P4	Port 4 DMA Source Address	0C0H	0FFH
SARL0	Low Byte/	0A2H	I
SARH0	Hi Byte/ Channel 0	0A3H	I
SARL1	Low Byte/	0B2H	I
SARH1	Hi Byte/ Channel 1	0B3H	I
*SLCON	Slave Control	0E8H	04H
SSTAT	Slave Status	0E9H	08FH

I = Indeterminate

The SFRs marked with an asterisk (*) are both bit- and byte- addressable. The functions of the SFRs are as follows:

Miscellaneous Special Function Register Description

80C51 SFRs

ACCUMULATOR

ACC is the Accumuator SFR. The mnemonics for accumulator-specific instructions, however, refer to the accumulator simply as A.

B REGISTER

The B SFR is used during multiply and divide operations. For other instructions it can be treated as another scratch pad regster.

PROGRAM STATUS WORD

The PSW SFR contains program status information as detailed in Table 12.

STACK POINTER

The Stack Pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

DATA POINTER

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

PORTS 0 TO 4

P0, P1, P2, P3 and P4 are the SFR latches of Ports 0, 1, 2, 3 and 4, respectively.

SERIAL DATA BUFFER

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

TIMER/COUNTER SFR

Register pairs (TH0, TL0), and (TH1, TL1) are the 16-bit counting registers for Timer/Counters 0 and 2.

POWER CONTROL SFR (PCON)

The PCON Register (Table 13) controls the power down and idle modes in the UPI-452, as well as providing the ability to double the Serial Channel baud rate. There are also two general purpose flag bits available to the user. Bits 5 and 6 are used to set the HOLD/HOLD Acknowledge mode (see "General Purpose DMA Channels" section), and bit 4 is not used.

Table 12. Program Status Word

Symbolic Address									Physical Address
PSW	CY	AC	FO	RS1	RS0	OV	—	P	0D0H
	(MSB)							(LSB)	

Symbol	Position	Name
CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry (For BCD operations)
F0	PSW.5	Flag 0 (user assignable)
RS1	PSW.4	Register Bank Select bit 1*
RS0	PSW.3	Register Bank Select bit 0*
OV	PSW.2	Overflow Flag
—	PSW.1	(reserved)
P	PSW.0	Parity Flag

*(RS1, RS0) enable internal RAM register banks as follows:

RS1	RS0	Internal RAM Register Bank
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

Table 13. PCON Special Function Register

Symbolic Address									Physical Address
PCON	SMOD	ARB	REQ	—	GF1	GF0	PD	IDL	087H
	(MSB)							(LSB)	

Symbol	Position	Function
SMOD	PCON7	Double Baud rate bit. When set to a 1, the baud rate is doubled when the serial port is being used in either Mode 1, 2 or 3.
ARB	PCON6	HLD/HLDA Arbiter control bit *
REQ	PCON5	HLD/HLDA Requestor control bit *
—	PCON4	(reserved)
GF1	PCON3	General-purpose flag bit
GF0	PCON2	General-purpose flag bit
PD	PCON1	Power Down bit. Setting this bit activates power down operation.
IDL	PCON0	Idle Mode bit. Setting this bit activates idle mode operation.

*See "Ext. Memory DMA" description.

NOTE:
If 1's are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (000X0000).

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . 0°C to 70°C†
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin to V_{SS} . . . . . −0.5V to V_{CC} + 0.5V
Voltage on V_{CC} to V_{SS} . . . . . −0.5V to +6.5V
Power Dissipation 1.0W**
V_{CC}/V_{PP} Supply Voltage with Respect to Ground During Programming . . . . . −0.6V to +14.0V

Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS T_A = 0°C to 70°C; V_{CC} = 5V ±10%; V_{SS} = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage (except XTAL1, RST)	2.0	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage (XTAL1, RST)	3.9	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage (Ports 1, 2, 3, 4)		0.45	V	I_{OL} = 1.6 mA (Note 1)
V_{OL1}	Output Low Voltage (except Ports 1, 2, 3, 4)		0.45	V	I_{OL} = 3.2 mA (Note 1)
V_{OH}	Output High Voltage (Ports 1, 2, 3, 4)	2.4		V	I_{OH} = −60 μA, V_{CC} = 5V ±10%
		0.75 V_{CC}		V	I_{OH} = −25 μA
		0.9 V_{CC}		V	I_{OH} = −10 μA
V_{OH1}	Output High Voltage (except Ports 1, 2, 3, 4 and Host Interface (Slave) Port)	2.4		V	I_{OH} = −400 μA, V_{CC} = 5V ±10%
		0.75 V_{CC}		V	I_{OH} = −150 μA
		0.9 V_{CC}		V	I_{OH} = −40 μA (Note 2)
V_{OH2}	Output High Voltage (Host Interface (Slave) Port)	2.4		V	I_{OH} = −400 μA, V_{CC} = 5V ±10%
		3.0		V	I_{OH} = 1 mA
		V_{CC} − 0.4		V	I_{OH} = −10 μA
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3, 4)		−50	μA	V_{IN} = 0.45V
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, 3, 4)		−650	μA	V_{IN} = 2V

† Ambient Temperature under Bias for 87C452P is 0°C to 50°C.

D.C. CHARACTERISTICS T_A = 0°C to 70°C; V_{CC} = 5V ±10%; V_{SS} = 0V (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
I_{LI}	Input Leakage Current (except Ports 1, 2, 3, 4)		±10	μA	$0.45V < V_{IN} < V_{CC}$
I_{OZ}	Output Leakage Current (except Ports 1, 2, 3, 4)		±10	μA	$0.45V < V_{OUT} < V_{CC}$
I_{CC_1}	Operating Current (Note 6)		15	mA	V_{CC} = 5.5V, 16 MHz
I_{CC}	Operating Current (Note 7)		50	mA	V_{CC} = 5.5V, 16 MHz (Note 4)
I_{CCI}	Idle Mode Current		25	mA	V_{CC} = 5.5V, 16 MHz (Note 5)
I_{PD}	Power Down Current		100	μA	V_{CC} = 2V (Note 3)
RRST	Reset Pulldown Resistor	50	150	KΩ	
CIO	Pin Capacitance		20	pF	1 MHz, T_A = 25°C (sampled, not tested on all parts)

NOTES:
1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OLS} of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
2. Capacitive loading on Ports 0 and 2 may cause the V_{OH} on ALE and PSEN to momentarily fall before the 0.9 V_{CC} specification when the address bits are stabilizing.
3. Power DOWN I_{CC} is measured with all output pins disconnected; EA = Port 0 = V_{CC}; XTAL2 N.C.; RST = V_{SS}; DB = V_{CC}; $\overline{WR}$ = $\overline{RD}$ = $\overline{DACK}$ = $\overline{CS}$ = A0 = A1 = A2 = V_{CC}. Power Down Mode is not supported on the 87C452P.
4. I_{CC} is measured with all output pins disconnected; XTAL1 driven with TCLCH, TCHCL = 5 ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} − 0.5V; XTAL2 N.C.; EA = RST = Port 0 = V_{CC}; $\overline{WR}$ = $\overline{RD}$ = $\overline{DACK}$ = $\overline{CS}$ = A0 = A1 = A2 = V_{CC}. I_{CC} would be slightly higher if a crystal oscillator is used.
5. Idle I_{CC} is measured with all output pins disconnected; XTAL1 driven with TCLCH, TCHCL = 5 ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} − 0.5V; XTAL2 N.C.; Port 0 = V_{CC}; EA = RST = V_{SS}; $\overline{WR}$ = $\overline{RD}$ = $\overline{DACK}$ = $\overline{CS}$ = A0 = A1 = A2 = V_{CC}.
6. 87C452P Piggyback EPROM only.
7. 80C452 and 83C452 only.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for:

A: Address.
C: Clock.
D: Input data.
H: Logic level HIGH.
I: Instruction (program memory contents).
L: Logic level LOW, or ALE.
P: PSEN.
Q: Output data.
R: READ signal.
T: Time.
V: Valid.
W: WRITE signal.
X: No longer a valid logic level.
Z: Float.

EXAMPLE

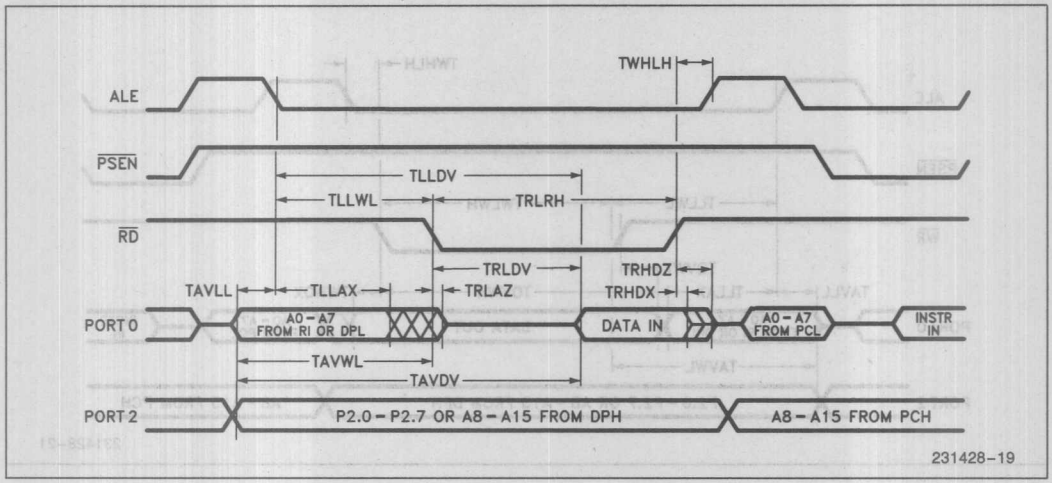
TAVLL = Time for Address Valid to ALE Low.
TLLPL = Time for ALE Low to $\overline{PSEN}$ Low.

A.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$, Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF, Load Capacitance for All Other Outputs = 80 pF

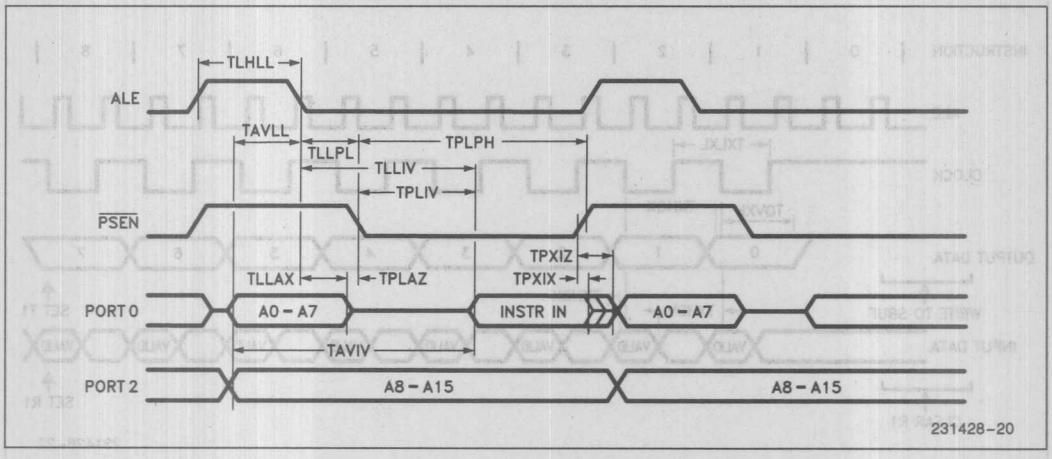
EXTERNAL PROGRAM AND DATA MEMORY CHARACTERISTICS

Symbol	Parameter	16 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency	3.5	16			MHz
TLHLL	ALE Pulse Width	85		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	25		TCLCL−55		ns
TLLAX	Address Hold after ALE Low	28		TCLCL−35		ns
TLLIV	ALE Low to Valid Instr In		185		4TCLCL−100	ns
TLLPL	ALE Low to PSEN Low	22		TCLCL−40		ns
TPLPH	PSEN Pulse Width	142		3TCLCL−45		ns
TPLIV	PSEN Low to Valid Instr In		110		3TCLCL−105	ns
TPXIX	Input Instr Hold after PSEN	0		0		ns
TPXIZ	Input Instr Float after PSEN		57		TCLCL−25	ns
TAVIV	Address to Valid Instr In		225		5TCLCL−105	ns
TPLAZ	PSEN Low to Address Float		10		10	ns
TRLRH	RD Pulse Width	275		6TCLCL−100		ns
TWLWH	WR Pulse Width	275		6TCLCL−100		ns
TRLDV	RD Low to Valid Data In		148		5TCLCL−165	ns
TRHDX	Data Hold after RD	0		0		ns
TRHDZ	Data Float after RD				2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		350		8TCLCL−150	ns
TAVDV	Address to Valid Data In		398		9TCLCL−165	ns
TLLWL	ALE Low to RD or WR Low	137	237	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address Valid to Read or Write Low	120		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition	2		TCLCL−60		ns
TWHQX	Data Hold after $\overline{WR}$	12		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	23	103	TCLCL−40	TCLCL+40	ns
TQVWH	Data Valid to $\overline{WR}$ (Setup Time)	288		7TCLCL−150		ns

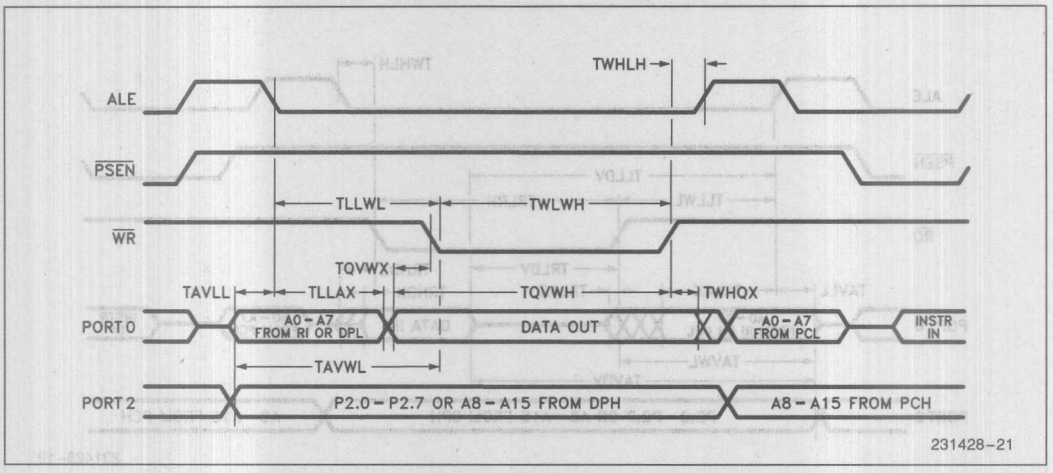
EXTERNAL DATA MEMORY READ CYCLE



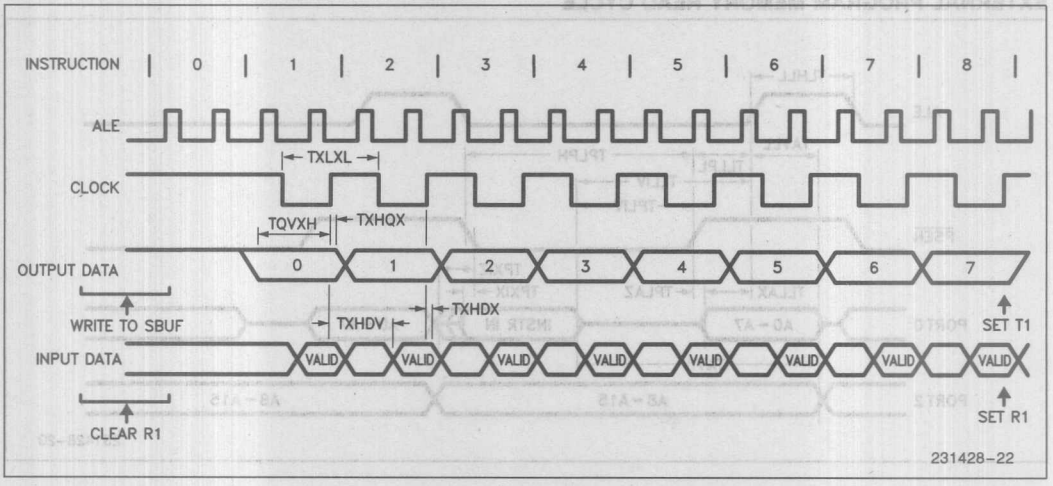
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE



SHIFT REGISTER MODE TIMING WAVEFORMS



EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	16	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

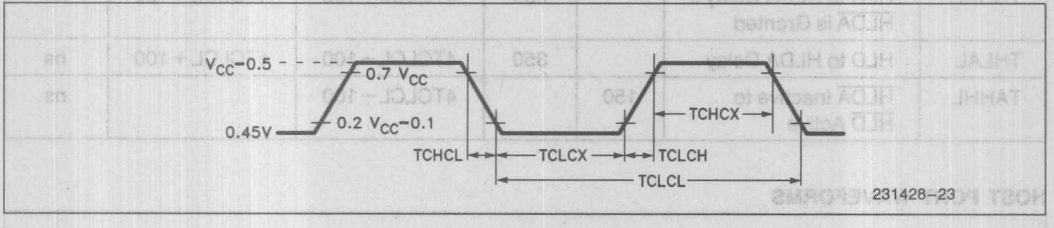
NOTE:
External clock timings are sampled, not tested on all parts.

SERIAL PORT TIMING—SHIFT REGISTER MODE

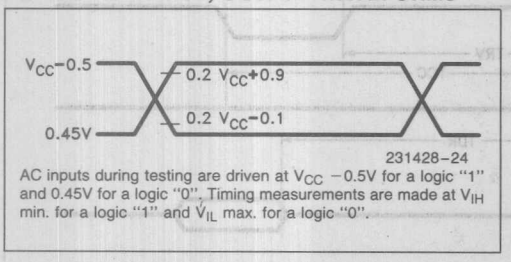
Test Conditions: T_A = 0°C to 70°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance = 80 pF

Symbol	Parameter	16 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	750		12TCLCL		ns
TQVXH	Output Data Setup to Clock Rising Edge	492		10TCLCL−133		ns
TXHQX	Output Data Hold after Clock Rising Edge	8		2TCLCL−117		ns
TXHDX	Input Data Hold after Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		492		10TCLCL−133	ns

EXTERNAL CLOCK DRIVE WAVEFORM

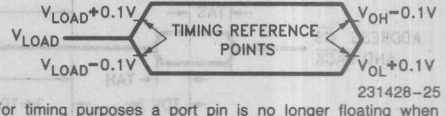


AC TESTING INPUT, OUTPUT WAVEFORMS



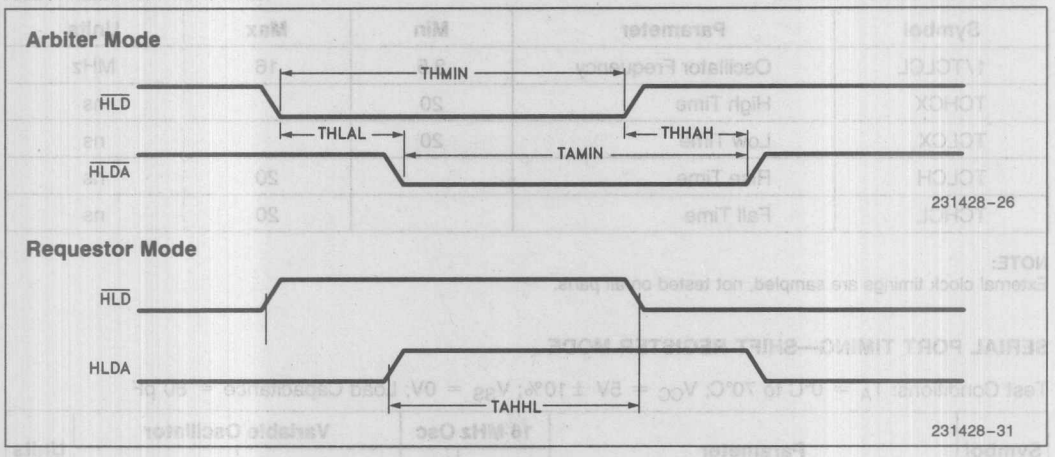
AC inputs during testing are driven at V_{CC} −0.5V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at V_{IH} min. for a logic "1" and V_{IL} max. for a logic "0".

FLOAT WAVEFORMS



For timing purposes a port pin is no longer floating when a 100 mV change from load voltage occurs, and begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OL}/I_{OH} \geq \pm 20$ mA.

HLD/HLDA WAVEFORMS

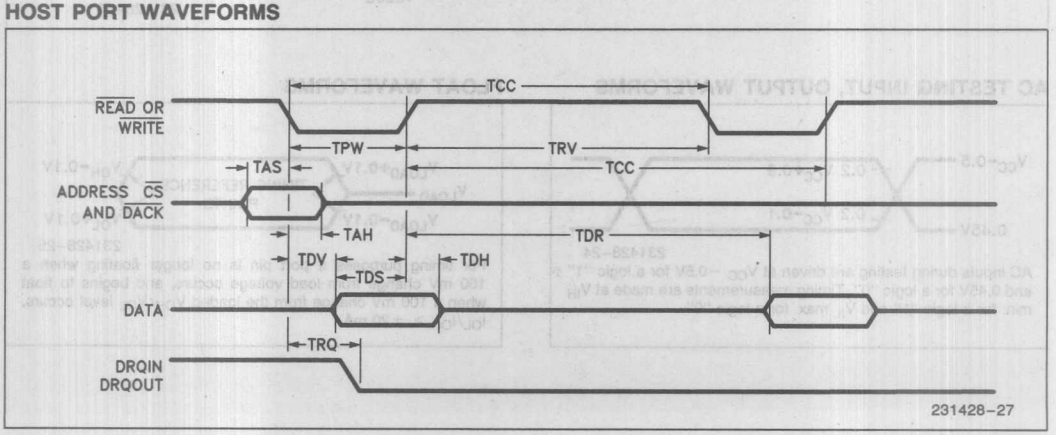


HLD/HLDA TIMINGS

Test Conditions: T_A = 0°C to +70°C; V_{CC} = 5V ±10%, V_{SS} = 0V; Load Capacitance = 80 pF

Symbol	Parameter	16 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
THMIN	$\overline{HLD}$ Pulse Width	350		4TCLCL+100		ns
TAMIN	$\overline{HLDA}$ Pulse Width	350		4TCLCL+100		ns
THHAH	$\overline{HLD}$ to $\overline{HLDA}$ Delay if $\overline{HLDA}$ is Granted		350	4TCLCL−100	4TCLCL+100	ns
THLAL	HLD to HLDA Delay		350	4TCLCL−100	4TCLCL+100	ns
TAHHL	$\overline{HLDA}$ Inactive to $\overline{HLD}$ Active	150		4TCLCL−100		ns

HOST PORT WAVEFORMS



HOST PORT TIMINGS

Test Conditions: T_A = 0°C to 70°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance = 80 pF

Symbol	Parameter	16 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TCC	Cycle Time	375		6TCLCL		ns
TPW	Command Pulse Width	100		100		ns
TRV	Recovery Time	60		60		ns
TAS	Address Setup Time	5		5		ns
TAH	Address Hold Time	30		30		ns
TDS	Write Data Setup Time	30		30		ns
TDHW	Write Data Hold Time	5		5		ns
TDHR	Read Data Hold Time	5	40	5	40	ns
TDV	$\overline{READ}$ Active to Read Data Valid Delay	85		92 ns on 12 MHz Part		ns
TDR	$\overline{WRITE}$ Inactive to Read Data Valid Delay (Applies only to Host Control SFR)		300		4.8TCLCL	ns
TRQ	$\overline{READ}$ or $\overline{WRITE}$ Active to DRQIN or DRQOUT Delay		150		150	ns

PROGRAMMING MODES

Caution: Exceeding 14V on V_{PP} will permanently damage the device.

Initially, all bits of the EPROM are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be present in the data word. The only way to change a "0" to a "1" is by ultraviolet light exposure (Cerdip EPROMs).

This device is in the programming mode when V_{PP} is raised to its programming voltage and ALE/$\overline{PGM}$ are both at TTL-low. The data to be programmed is applied 8 bits in parallel to the Port 0 pins. The levels required for the address and data inputs are TTL. The address is applied to Port 1 and 2.

Program Verify

A verify should be performed on the programmed bits to determine that they have been correctly programmed. The verify is performed with $\overline{OE}$ at V_{IL}, $\overline{PGM}$ at V_{IH} and V_{PP} and V_{CC} at their programming voltages.

inteligent Identifier™ Mode

inteligent Identifier™ Mode is not supported on the 87C452P piggyback EPROM device.

ERASURE CHARACTERISTICS

The erasure characteristics are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å–4000Å range. Data shows that constant exposure to room level fluorescent lighting could erase the EPROM in approximately three years, while it would take approximately one week to cause erasure when exposed to direct sunlight. If the EPROM is to be exposed to these types of lighting conditions for extended periods of time, opaque labels should be placed over the window to prevent unintentional erasure.

The recommended erasure procedure is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity × exposure time) for erasure should be a minimum of fifteen (15) Wsec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000 μW/cm^2 power rating. The EPROM should be placed within one inch of the lamp tubes during erasure. The maximum integrated dose an EPROM can be exposed to without damage is 7258 Wsec/cm^2 (1 week @ 12000 μW/cm^2). Exposure of the EPROM to high intensity UV light for longer periods may cause permanent damage.

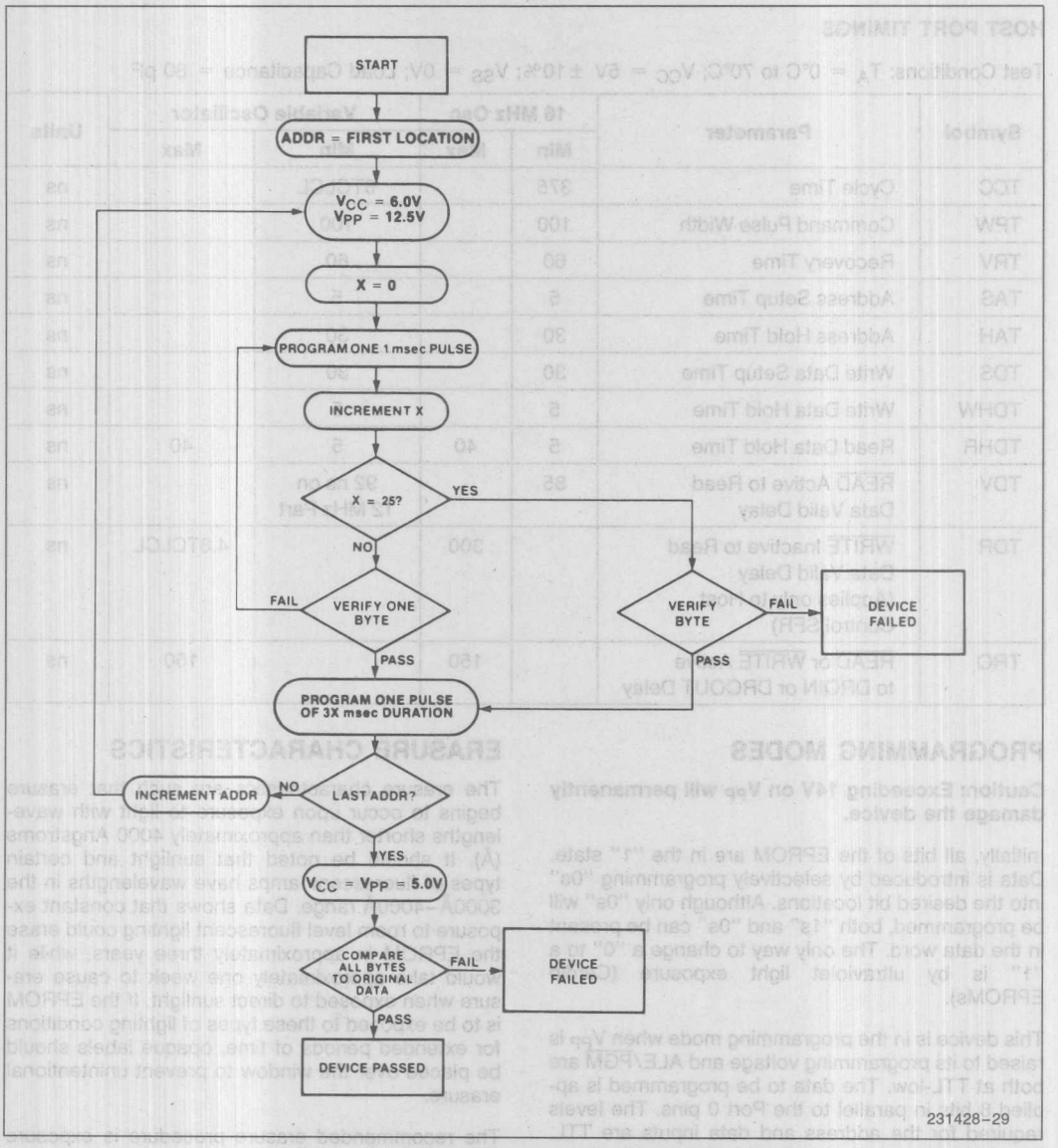


Figure 4. int$_e$ligent Programming™ Flowchart

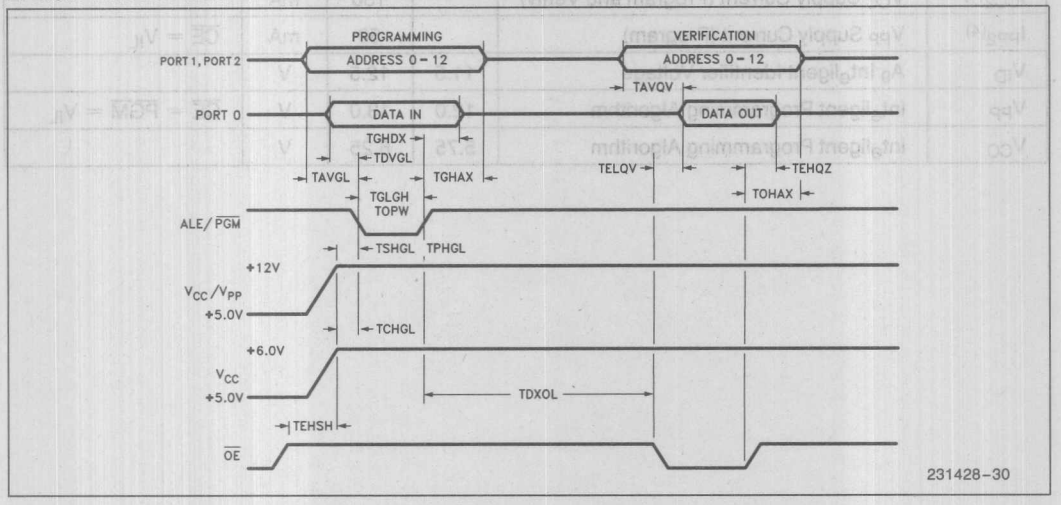
int$_e$ligent Programming™ Algorithm

The int$_e$ligent Programming Algorithm, a standard in the industry for the past few years, is required for all of Intel's 12.5V DERDEP EPROMs. Plastic amd PLCC EPROMs may also be programmed using this method. A flowchart of the int$_e$ligent Programming Algorithm is shown in Figure 4.

The int$_e$ligent Programming Algorithm utilizes two different pulse types: initial and overprogram. The duration of the initial $\overline{PGM}$ pulse(s) is one millisecond, which will then be followed by a longer overprogram pulse of length 3X ms. X is an iteration counter and is equal to the number of the initial one millisecond pulses applied to a particular location, before a correct verify occurs. Up to 25 one-millisecond pulses per byte are provided for before the overprogram pulse is applied.

The entire sequence of program pulses and byte verifications is performed at V_{CC} = 6.0V and V_{PP} = 12.5V. When the int$_e$ligent Programming cycle has been completed, all bytes should be compared to the original data with V_{CC} = V_{PP} = 5.0V.

EPROM PROGRAMMING AND VERIFICATION WAVEFORMS



A.C. PROGRAMMING CHARACTERISTICS

T_A = 25°C ±5°C (see EPROM PROGRAMMING D.C. CHARACTERISTICS for V_{CC} and V_{PP} Voltages)

Symbol	Parameter	12 MHz OSC			Test Conditions*(1)
		Min	Max	Unit	
TAVGL	Address Setup to $\overline{PGM}$		2.9	µs	
TDXOL	$\overline{OE}$ Setup from Data Float	2.0		µs	
TDVGL	Data Setup to $\overline{PGM}$	3.8		µs	
TOHAX	Address Hold after $\overline{OE}$	0		µs	
TGHDX	Data Hold after $\overline{PGM}$	2.0		ns	
TEHQZ	Data Float after $\overline{OE}$	0	1.6	µs	(Note 3)
TOHAX	Address Hold after $\overline{PGM}$	0		µs	
TPHGL	V_{PP} Setup to $\overline{PGM}$	2.0		µs	
TCHGL	V_{CC} Setup to $\overline{PGM}$	2.0		µs	
TEHSH	$\overline{OE}$ Setup to V_{PP} and V_{CC}	2.0		µs	
TGLGH	$\overline{PGM}$ Pulse Width	0.95	1.05	ms	int$_e$ligent Programming
TAVQV	Address to Data Valid		3.0	µs	
TOPW	$\overline{PGM}$ Overprogram Pulse Width	2.85	78.75	ms	(Note 2)
TELQV	Data Valid from $\overline{OE}$		2.0	µs	

NOTES:
1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
2. The length of the overprogram pulse may vary from 2.85 ms to 78.75 ms as a function of the iteration counter value X (int$_e$ligent Programming Algorithm only).
3. This parameter is only sampled and is not 100% tested. Output Float is defined as the point where data is no longer driven—see timing diagram.
4. The maximum current value is with Port 0 unloaded.

D.C. PROGRAMMING CHARACTERISTICS $T_A = 25°C \pm 5\%$

Symbol	Parameter	Limits			Test Conditions*(1)
		Min	Max	Unit	
I_{CC2}(4)	V_{CC} Supply Current (Program and Verify)		150	mA	
I_{PP2}(4)	V_{PP} Supply Current (Program)		50	mA	$\overline{CE} = V_{IL}$
V_{ID}	A_9 Int$_e$ligent Identifier Voltage	11.5	12.5	V	
V_{PP}	int$_e$ligent Programming Algorithm	12.0	13.0	V	$\overline{CE} = \overline{PGM} = V_{IL}$
V_{CC}	int$_e$ligent Programming Algorithm	5.75	6.25	V	

NOTE

November 1987

Using the Intel MCS®-51 Boolean Processing Capabilities

JOHN WHARTON
MICROCONTROLLER APPLICATIONS

Order Number: 203830-001

1.0 INTRODUCTION

The Intel microcontroller family now has three new members: the Intel® 8031, 8051, and 8751 single-chip microcomputers. These devices, shown in Figure 1, will allow whole new classes of products to benefit from recent advances in Integrated Electronics. Thanks to Intel's new HMOS technology, they provide larger program and data memory spaces, more flexible I/O and peripheral capabilities, greater speed, and lower system cost than any previous-generation single-chip microcomputer.

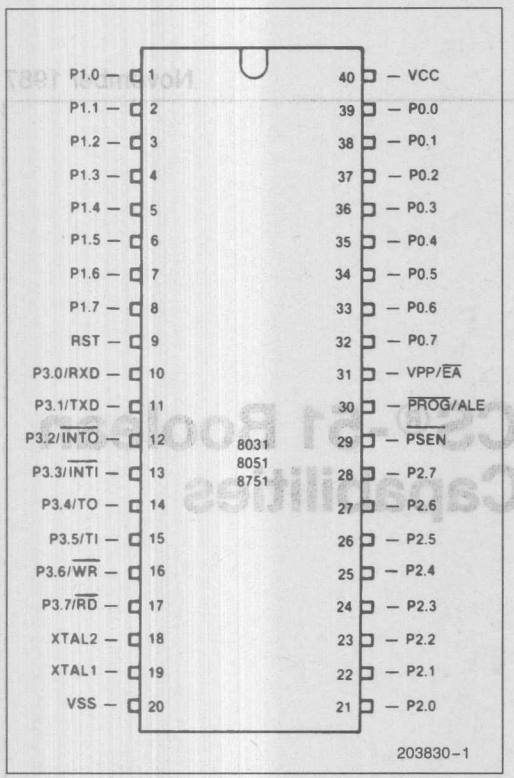


Figure 1. 8051 Family Pinout Diagram

Table 1 summarizes the quantitative differences between the members of the MCS®-48 and 8051 families. The 8751 contains 4K bytes of EPROM program memory fabricated on-chip, while the 8051 replaces the EPROM with 4K bytes of lower-cost mask-programmed ROM. The 8031 has no program memory on-chip; instead, it accesses up to 64K bytes of program memory from external memory. Otherwise, the three new family members are identical. Throughout this Note, the term "8051" will represent all members of the 8051 Family, unless specifically stated otherwise.

The CPU in each microcomputer is one of the industry's fastest and most efficient for numerical calculations on byte operands. But controllers often deal with bits, not bytes: in the real world, switch contacts can only be open or closed, indicators should be either lit or dark, motors are either turned on or off, and so forth. For such control situations the most significant aspect of the MCS®-51 architecture is its complete hardware support for one-bit, or *Boolean* variables (named in honor of Mathematician George Boole) as a separate data type.

The 8051 incorporates a number of special features which support the direct manipulation and testing of individual bits and allow the use of single-bit variables in performing logical operations. Taken together, these features are referred to as the MCS-51 *Boolean Processor*. While the bit-processing capabilities alone would be adequate to solve many control applications, their true power comes when they are used in conjunction with the microcomputer's byte-processing and numerical capabilities.

Many concepts embodied by the Boolean Processor will certainly be new even to experienced microcomputer system designers. The purpose of this Application Note is to explain these concepts and show how they are used.

For detailed information on these parts refer to the **Intel Microcontroller Handbook,** order number 210918. The instruction set, assembly language, and use of the 8051 assembler (ASM51) are further described in the **MCS®-51 Macro Assembler User's Guide for DOS Systems,** order number 122753.

Table 1. Features of Intel's Single-Chip Microcomputers

EPROM Program Memory	ROM Program Memory	External Program Memory	Program Memory (Int/Max)	Data Memory (Bytes)	Instr. Cycle Time	Input/ Output Pins	Interrupt Sources	Reg. Banks
8748	8048	8035	1K 4K	64	2.5 μs	27	2	2
—	8049	8039	2K 4K	128	1.36 μs	27	2	2
8751	8051	8031	4K 64K	128	1.0 μs	32	5	4

2.0 BOOLEAN PROCESSOR OPERATION

The Boolean Processing capabilities of the 8051 are based on concepts which have been around for some time. Digital computer systems of widely varying designs all have four functional elements in common (Figure 2):

- a central processor (CPU) with the control, timing, and logic circuits needed to execute stored instructions:
- a memory to store the sequence of instructions making up a program or algorithm:
- data memory to store variables used by the program:
  and
- some means of communicating with the outside world.

The CPU usually includes one or more accumulators or special registers for computing or storing values during program execution. The instruction set of such a processor generally includes, at a minimum, operation classes to perform arithmetic or logical functions on program variables, move variables from one place to another, cause program execution to jump or conditionally branch based on register or variable states, and instructions to call and return from subroutines. The program and data memory functions sometimes share a single memory space, but this is not always the case. When the address spaces are separated, program and data memory need not even have the same basic word width.

A digital computer's flexibility comes in part from combining simple fast operations to produce more complex (albeit slower) ones, which in turn link together eventually solving the problem at hand. A four-bit CPU executing multiple precision subroutines can, for example, perform 64-bit addition and subtraction. The subroutines could in turn be building blocks for floating-point multiplication and division routines. Eventually, the four-bit CPU can simulate a far more complex "virtual" machine.

In fact, *any* digital computer with the above four functional elements can (given time) complete *any* algorithm (though the proverbial room full of chimpanzees at word processors might first re-create Shakespeare's classics and this Application Note)! This fact offers little consolation to product designers who want programs to run as quickly as possible. By definition, a real-time control algorithm *must* proceed quickly enough to meet the preordained speed constraints of other equipment.

One of the factors determining how long it will take a microcomputer to complete a given chore is the number of instructions it must execute. What makes a given computer architecture particularly well- or poorly-suited for a class of problems is how well its instruction set matches the tasks to be performed. The better the "primitive" operations correspond to the steps taken by the control algorithm, the lower the number of instructions needed, and the quicker the program will run. All else being equal, a CPU supporting 64-bit arithmetic directly could clearly perform floating-point math faster than a machine bogged-down by multiple-precision subroutines. In the same way, direct support for bit manipulation naturally leads to more efficient programs handling the binary input and output conditions inherent in digital control problems.

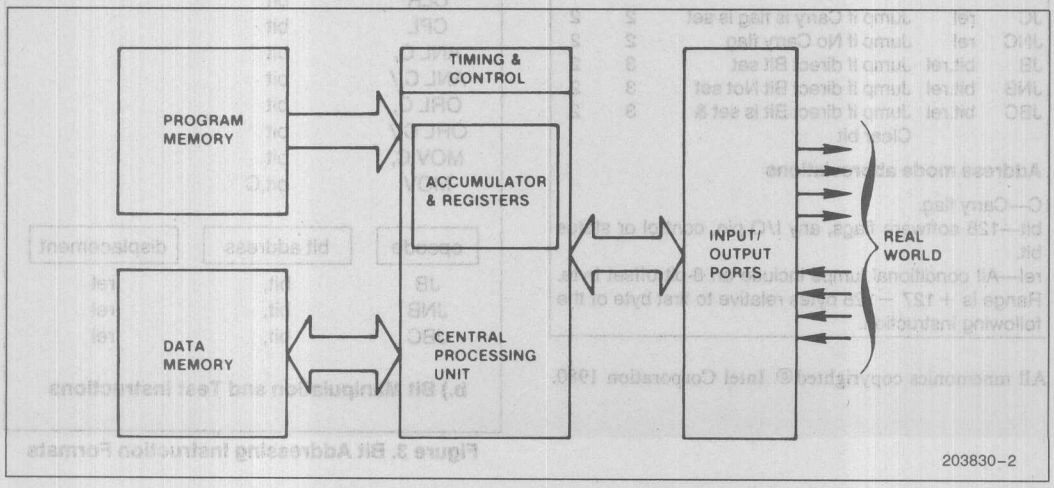


Figure 2. Block Diagram for Abstract Digital Computer

Processing Elements

The introduction stated that the 8051's bit-handling capabilities alone would be sufficient to solve some control applications. Let's see how the four basic elements of a digital computer—a CPU with associated registers, program memory, addressable data RAM, and I/O capability—relate to Boolean variables.

CPU. The 8051 CPU incorporates special logic devoted to executing several bit-wide operations. All told, there are 17 such instructions, all listed in Table 2. Not shown are 94 other (mostly byte-oriented) 8051 instructions.

Program Memory. Bit-processing instructions are fetched from the same program memory as other arithmetic and logical operations. In addition to the instructions of Table 2, several sophisticated program control features like multiple addressing modes, subroutine nesting, and a two-level interrupt structure are useful in structuring Boolean Processor-based programs.

Table 2. MCS-51™ Boolean Processing Instruction Subset

Mnemonic		Description	Byte	Cyc
SETB	C	Set Carry flag	1	1
SETB	bit	Set direct Bit	2	1
CLR	C	Clear Carry flag	1	1
CLR	bit	Clear direct bit	2	1
CPL	C	Complement Carry flag	1	1
CPL	bit	Complement direct bit	2	1
MOV	C.bit	Move direct bit to Carry flag	2	1
MOV	bit.C	Move Carry flag to direct bit	2	2
ANL	C.bit	AND direct bit to Carry flag	2	2
ANL	C.bit	AND complement of direct bit to Carry flag	2	2
ORL	C.bit	OR direct bit to Carry flag	2	2
ORL	C.bit	OR complement of direct bit to Carry flag	2	2
JC	rel	Jump if Carry is flag is set	2	2
JNC	rel	Jump if No Carry flag	2	2
JB	bit.rel	Jump if direct Bit set	3	2
JNB	bit.rel	Jump if direct Bit Not set	3	2
JBC	bit.rel	Jump if direct Bit is set & Clear bit	3	2

Address mode abbreviations

C—Carry flag.
bit—128 software flags, any I/O pin, control or status bit.
rel—All conditional jumps include an 8-bit offset byte. Range is +127 −128 bytes relative to first byte of the following instruction.

All mnemonics copyrighted© Intel Corporation 1980.

Boolean instructions are one, two, or three bytes long, depending on what function they perform. Those involving only the carry flag have either a single-byte opcode or an opcode followed by a conditional-branch destination byte (Figure 3a). The more general instructions add a "direct address" byte after the opcode to specify the bit affected, yielding two or three byte encodings (Figure 3b). Though this format allows potentially 256 directly addressable bit locations, not all of them are implemented in the 8051 family.

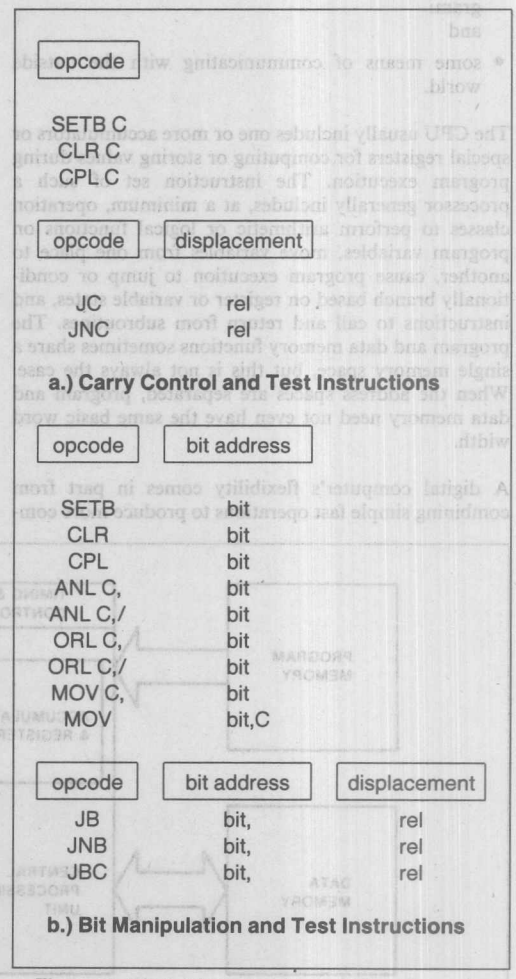


Figure 3. Bit Addressing Instruction Formats

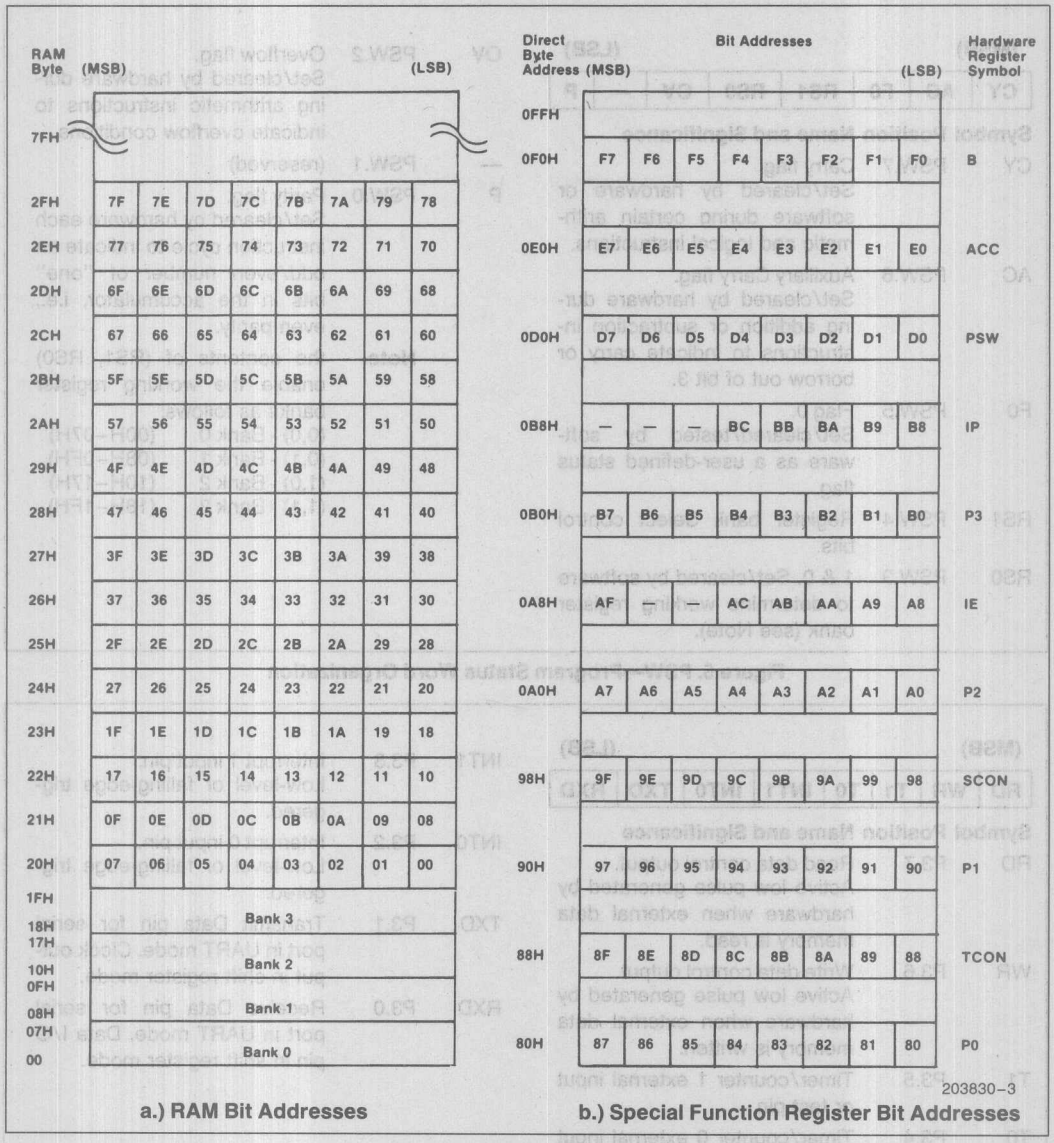


Figure 4. Bit Address Maps

Data Memory. The instructions in Figure 3b can operate directly upon 144 general purpose bits forming the Boolean processor "RAM." These bits can be used as software flags or to store program variables. Two operand instructions use the CPU's carry flag ("C") as a special one-bit register: in a sense, the carry is a "Boolean accumulator" for logical operations and data transfers.

Input/Output. All 32 I/O pins can be addressed as individual inputs, outputs, or both, in any combination. Any pin can be a control strobe output, status (Test) input, or serial I/O link implemented via software. An additional 33 individually addressable bits reconfigure, control, and monitor the status of the CPU and all on-chip peripheral functions (timer counters, serial port modes, interrupt logic, and so forth).

(MSB)							(LSB)
CY	AC	F0	RS1	RS0	OV	—	P

Symbol	Position	Name and Significance
CY	PSW.7	Carry flag. Set/cleared by hardware or software during certain arithmetic and logical instructions.
AC	PSW.6	Auxiliary Carry flag. Set/cleared by hardware during addition or subtraction instructions to indicate carry or borrow out of bit 3.
F0	PSW.5	Flag 0. Set/cleared/tested by software as a user-defined status flag.
RS1	PSW.4	Register bank Select control bits.
RS0	PSW.3	1 & 0. Set/cleared by software to determine working register bank (see Note).
OV	PSW.2	Overflow flag. Set/cleared by hardware during arithmetic instructions to indicate overflow conditions.
—	PSW.1	(reserved)
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the accumulator, i.e., even parity.

Note- the contents of (RS1, RS0) enable the working register banks as follows:

(0,0) - Bank 0 (00H–07H)
(0,1) - Bank 1 (08H–0FH)
(1,0) - Bank 2 (10H–17H)
(1,1) - Bank 3 (18H–1FH)

Figure 5. PSW—Program Status Word Organization

(MSB)							(LSB)
RD	WR	T1	T0	INT1	INT0	TXD	RXD

Symbol	Position	Name and Significance
RD	P3.7	Read data control output. Active low pulse generated by hardware when external data memory is read.
WR	P3.6	Write data control output. Active low pulse generated by hardware when external data memory is written.
T1	P3.5	Timer/counter 1 external input or test pin.
T0	P3.4	Timer/counter 0 external input or test pin.
INT1	P3.3	Interrupt 1 input pin. Low-level or falling-edge triggered.
INT0	P3.2	Interrupt 0 input pin. Low-level or falling-edge triggered.
TXD	P3.1	Transmit Data pin for serial port in UART mode. Clock output in shift register mode.
RXD	P3.0	Receive Data pin for serial port in UART mode. Data I/O pin in shift register mode.

Figure 6. P3—Alternate I/O Functions of Port 3

Direct Bit Addressing

The most significant bit of the direct address byte selects one of two groups of bits. Values between 0 and 127 (00H and 7FH) define bits in a block of 32 bytes of on-chip RAM, between RAM addresses 20H and 2FH (Figure 4a). They are numbered consecutively from the lowest-order byte's lowest-order bit through the highest-order byte's highest-order bit.

Bit addresses between 128 and 255 (80H and 0FFH) correspond to bits in a number of special registers, mostly used for I/O or peripheral control. These positions are numbered with a different scheme than RAM: the five high-order address bits match those of the register's own address, while the three low-order bits identify the bit position within that register (Figure 4b).

Notice the column labeled "Symbol" in Figure 5. Bits with special meanings in the PSW and other registers have corresponding symbolic names. General-purpose (as opposed to carry-specific) instructions may access the carry like any other bit by using the mnemonic CY in place of C, P0, P1, P2, and P3 are the 8051's four I/O ports: secondary functions assigned to each of the eight pins of P3 are shown in Figure 6.

Figure 7 shows the last four bit addressable registers. TCON (Timer Control) and SCON (Serial port Control) control and monitor the corresponding peripherals, while IE (Interrupt Enable) and IP (Interrupt Priority) enable and prioritize the five hardware interrupt sources. Like the reserved hardware register addresses, the five bits not implemented in IE and IP should not be accessed: they can *not* be used as software flags.

Addressable Register Set. There are 20 special function registers in the 8051, but the advantages of bit addressing only relate to the 11 described below. Five potentially bit-addressable register addresses (0C0H, 0C8H, 0D8H, 0E8H, & 0F8H) are being reserved for possible future expansion in microcomputers based on the MCS-51 architecture. Reading or writing non-existent registers in the 8051 series is pointless, and may cause unpredictable results. Byte-wide logical operations can be used to manipulate bits in all *non*-bit addressable registers and RAM.

(MSB)							(LSB)
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn timer/counter on/off.
TF0	TCON.5	Timer 0 overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn timer/counter on/off.
IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

a.) TCON—Timer/Counter Control/Status Register

(MSB)							(LSB)
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Symbol	Position	Name and Significance
SM0	SCON.7	Serial port Mode control bit 0. Set/cleared by software (see note).
SM1	SCON.6	Serial port Mode control bit 1. Set/cleared by software (see note).
SM2	SCON.5	Serial port Mode control bit 2. Set by software to disable reception of frames for which bit 8 is zero.
REN	SCON.4	Receiver Enable control bit. Set/cleared by software to enable/disable serial data reception.
TB8	SCON.3	Transmit Bit 8. Set/cleared by hardware to determine state of ninth data bit transmitted in 9-bit UART mode.
RB8	SCON.2	Receive Bit 8. Set/cleared by hardware to indicate state of ninth data bit received.
TI	SCON.1	Transmit Interrupt flag. Set by hardware when byte transmitted. Cleared by software after servicing.
RI	SCON.0	Receive Interrupt flag. Set by hardware when byte received. Cleared by software after servicing.
	Note-	the state of (SM0, SM1) selects: (0,0)—Shift register I/O expansion. (0,1)—8-bit UART, variable data rate. (1,0)—9-bit UART, fixed data rate. (1,1)—9-bit UART, variable data rate.

b.) SCON—Serial Port Control/Status Register

Figure 7. Peripheral Configuration Registers

(MSB) (LSB)

EA	—	—	ES	ET1	EX1	ET1	EX0

Symbol	Position	Name and Significance
EA	IE.7	Enable All control bit. Cleared by software to disable all interrupts, independent of the state of IE.4–IE.0.
—	IE.6	(reserved)
—	IE.5	
ES	IE.4	Enable Serial port control bit. Set/cleared by software to enable/disable interrupts from TI or RI flags.
ET1	IE.3	Enable Timer 1 control bit. Set/cleared by software to enable/disable interrupts from timer/counter 1.
EX1	IE.2	Enable External interrupt 1 control bit. Set/cleared by software to enable/disable interrupts from INT1.
ET0	IE.1	Enable Timer 0 control bit. Set/cleared by software to enable/disable interrupts from timer/counter 0.
EX0	IE.0	Enable External interrupt 0 control bit. Set/cleared by software to enable/disable interrupts from INT0.

c.) IE—Interrupt Enable Register

(MSB) (LSB)

—	—	—	PS	PT1	PX1	PT0	PX0

Symbol	Position	Name and Significance
—	IP.7	(reserved)
—	IP.6	(reserved)
—	IP.5	(reserved)
PS	IP.4	Serial port Priority control bit. Set/cleared by software to specify high/low priority interrupts for Serial port.
PT1	IP.3	Timer 1 Priority control bit. Set/cleared by software to specify high/low priority interrupts for timer/counter 1.
PX1	IP.2	External interrupt 1 Priority control bit. Set/cleared by software to specify high/low priority interrupts for INT1.
PT0	IP.1	Timer 0 Priority control bit. Set/cleared by software to specify high/low priority interrupts for timer/counter 0.
PX0	IP.0	External interrupt 0 Priority control bit. Set/cleared by software to specify high/low priority interrupts for INT0.

d.) IP—Interrupt Priority Control Register

Figure 7. Peripheral Configuration Registers (Continued)

The accumulator and B registers (A and B) are normally involved in byte-wide arithmetic, but their individual bits can also be used as 16 general software flags. Added with the 128 flags in RAM, this gives 144 general purpose variables for bit-intensive programs. The program status word (PSW) in Figure 5 is a collection of flags and machine status bits including the carry flag itself. Byte operations acting on the PSW can therefore affect the carry.

Instruction Set

Having looked at the bit variables available to the Boolean Processor, we will now look at the four classes of instructions that manipulate these bits. It may be helpful to refer back to Table 2 while reading this section.

State Control. Addressable bits or flags may be set, cleared, or logically complemented in one instruction cycle with the two-byte instructions SETB, CLR, and CPL. (The "B" affixed to SETB distinguishes it from the assembler "SET" directive used for symbol definition.) SETB and CLR are analogous to loading a bit with a constant: 1 or 0. Single byte versions perform the same three operations on the carry.

The MCS-51 assembly language specifies a bit address in any of three ways:

- by a number or expression corresponding to the direct bit address (0–255):

- by the name or address of the register containing the bit, the *dot operator* symbol (a period: "."), and the bit's position in the register (7–0):
- in the case of control and status registers, by the predefined assembler symbols listed in the first columns of Figures 5–7.

Bits may also be given user-defined names with the assembler "BIT" directive and any of the above techniques. For example, bit 5 of the PSW may be cleared by any of the four instructions.

```
USR_FLG BIT   PSW.5     ; User Symbol Definition
        ...   ...
        CLR   0D5H      ; Absolute Addressing
        CLR   PSW.5     ; Use of Dot Operator
        CLR   F0        ; Pre-Defined Assembler
                        ; Symbol
        CLR   USR_FLG   ; User-Defined Symbol
```

Data Transfers. The two-byte MOV instructions can transport any addressable bit to the carry in one cycle, or copy the carry to the bit in two cycles. A bit can be moved between two arbitrary locations via the carry by combining the two instructions. (If necessary, push and pop the PSW to preserve the previous contents of the carry.) These instructions can replace the multi-instruction sequence of Figure 8, a program structure appearing in controller applications whenever flags or outputs are conditionally switched on or off.

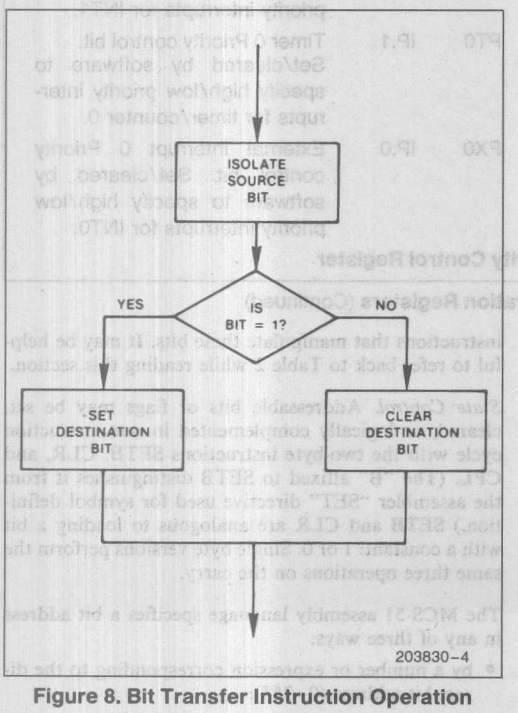


Figure 8. Bit Transfer Instruction Operation

Logical Operations. Four instructions perform the logical-AND and logical-OR operations between the carry and another bit, and leave the results in the carry. The instruction mnemonics are ANL and ORL; the absence or presence of a slash mark ("/") before the source operand indicates whether to use the positive-logic value or the logical complement of the addressed bit. (The source operand itself is never affected.)

Bit-test Instructions. The conditional jump instructions "JC rel" (Jump on Carry) and "JNC rel" (Jump on Not Carry) test the state of the carry flag, branching if it is a one or zero, respectively. (The letters "rel" denote relative code addressing.) The three-byte instructions "JB bit.rel" and "JNB bit.rel" (Jump on Bit and Jump on Not Bit) test the state of *any* addressable bit in a similar manner. A fifth instruction combines the Jump on Bit and Clear operations. "JBC bit.rel" conditionally branches to the indicated address, then clears the bit in the same two cycle instruction. This operation is the same as the MCS-48 "JTF" instructions.

All 8051 conditional jump instructions use program counter-relative addressing, and all execute in two cycles. The last instruction byte encodes a signed displacement ranging from −128 to +127. During execution, the CPU adds this value to the incremented program counter to produce the jump destination. Put another way, a conditional jump to the immediately following instruction would encode 00H in the offset byte.

A section of program or subroutine written using only relative jumps to nearby addresses will have the same machine code independent of the code's location. An assembled routine may be repositioned anywhere in memory, even crossing memory page boundaries, without having to modify the program or recompute destination addresses. To facilitate this flexibility, there is an unconditional "Short Jump" (SJMP) which uses relative addressing as well. Since a programmer would have quite a chore trying to compute relative offset values from one instruction to another, ASM51 automatically computes the displacement needed given only the destination address or label. An error message will alert the programmer if the destination is "out of range."

The so-called "Bit Test" instructions implemented on many other microprocessors simply perform the logical-AND operation between a byte variable and a constant mask, and set or clear a zero flag depending on the result. This is essentially equivalent to the 8051 "MOV C.bit" instruction. A second instruction is then needed to conditionally branch based on the state of the zero flag. This does *not* constitute abstract bit-addressing in the MCS-51 sense. A flag exists only as a field

within a register: to reference a bit the programmer must know and specify both the encompassing register and the bit's position therein. This constraint severely limits the flexibility of symbolic bit addressing and reduces the machine's code-efficiency and speed.

Interaction with Other Instructions. The carry flag is also affected by the instructions listed in Table 3. It can be rotated through the accumulator, and altered as a side effect of arithmetic instructions. Refer to the User's Manual for details on how these instructions operate.

Simple Instruction Combinations

By combining general purpose bit operations with certain addressable bits, one can "custom build" several hundred useful instructions. All eight bits of the PSW can be tested directly with conditional jump instructions to monitor (among other things) parity and overflow status. Programmers can take advantage of 128 software flags to keep track of operating modes, resource usage, and so forth.

The Boolean instructions are also the most efficient way to control or reconfigure peripheral and I/O registers. All 32 I/O lines become "test pins," for example, tested by conditional jump instructions. Any output pin can be toggled (complemented) in a single instruction cycle. Setting or clearing the Timer Run flags (TR0 and TR1) turn the timer/counters on or off; polling the same flags elsewhere lets the program determine if a timer is running. The respective overflow flags (TF0 and TF1) can be tested to determine when the desired period or count has elapsed, then cleared in preparation for the next repetition. (For the record, these bits are all part of the TCON register, Figure 7a. Thanks to symbolic bit addressing, the programmer only needs to remember the mnemonic associated with each function. In other words, don't bother memorizing control word layouts.)

In the MCS-48 family, instructions corresponding to some of the above functions require specific opcodes. Ten different opcodes serve to clear complement the software flags F0 and F1, enable/disable each interrupt, and start/stop the timer. In the 8051 instruction set, just three opcodes (SETB, CLR, CPL) with a direct bit address appended perform the same functions. Two test instructions (JB and JNB) can be combined with bit addresses to test the software flags, the 8048 I/O pins T0, T1, and INT, and the eight accumulator bits, replacing 15 more different instructions.

Table 4a shows how 8051 programs implement software flag and machine control functions associated with special opcodes in the 8048. In every case the MCS-51 solution requires the same number of machine cycles, and executes 2.5 times faster.

Table 3. Other Instructions Affecting the Carry Flag

Mnemonic		Description	Byte	Cyc
ADD	A,Rn	Add register to Accumulator	1	1
ADD	A,direct	Add direct byte to Accumulator	2	1
ADD	A,@Ri	Add indirect RAM to Accumulator	1	1
ADD	A,#data	Add immediate data to Accumulator	2	1
ADDC	A,Rn	Add register to Accumulator with Carry flag	1	1
ADDC	A,direct	Add direct byte to Accumulator with Carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry flag	1	1
ADDC	A,#data	Add immediate data to Acc with Carry flag	2	1
SUBB	A,Rn	Subtract register from Accumulator with borrow	1	1
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from Acc with borrow	1	1
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	1
MUL	AB	Multiply A & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1
RLC	A	Rotate Accumulator Left through the Carry flag	1	1
RRC	A	Rotate Accumulator Right through Carry flag	1	1
CJNE	A,direct.rel	Compare direct byte to Acc & Jump if Not Equal	3	2
CJNE	A,#data.rel	Compare immediate to Acc & Jump if Not Equal	3	2
CJNE	Rn,#data.rel	Compare immed to register & Jump if Not Equal	3	2
CJNE	@Ri,#data.rel	Compare immed to indirect & Jump if Not Equal	3	2

All mnemonics copyrighted © Intel Corporation 1980.

Table 4a. Contrasting 8048 and 8051 Bit Control and Testing Instructions

8048 Instruction		Bytes	Cycles	μSec	8x51 Instruction		Bytes	Cycles & μSec
Flag Control								
CLR	C	1	1	2.5	CLR	C	1	1
CPL	F0	1	1	2.5	CPL	F0	2	1
Flag Testing								
JNC	offset	2	2	5.0	JNC	rel	2	2
JF0	offset	2	2	5.0	JB	F0.rel	3	2
JB7	offset	2	2	5.0	JB	ACC.7.rel	3	2
Peripheral Polling								
JT0	offset	2	2	5.0	JB	T0.rel	3	2
JN1	offset	2	2	5.0	JNB	INT0.rel	3	2
JTF	offset	2	2	5.0	JBC	TF0.rel	3	2
Machine and Peripheral Control								
STRT	T	1	1	2.5	SETB	TR0	2	1
EN	I	1	1	2.5	SETB	EX0	2	1
DIS	TCNT1	1	1	2.5	CLR	ET0	2	1

Table 4b. Replacing 8048 Instruction Sequences with Single 8x51 Instructions

8048 Instruction		Bytes	Cycles	μSec	8051 Instruction		Bytes	Cycles & μSec
Flag Control								
Set carry								
CLR	C							
CPL	C	= 2	2	5.0	SETB	C	1	1
Set Software Flag								
CLR	F0							
CPL	F0	= 2	2	5.0	SETB	F0	2	1
Turn Off Output Pin								
ANL	P1.#0FBH	= 2	2	5.0	CLR	P1.2	2	1
Complement Output Pin								
IN	A.P1							
XRL	A.#04H							
OUTL	P1.A	= 4	6	15.0	CPL	P1.2	2	1
Clear Flag in RAM								
MOV	R0.#FLGADR							
MOV	A.@R0							
ANL	A.#FLGMASK							
MOV	@R0.A	= 6	6	15.0	CLR	USER_FLG	2	1

Table 4b. Replacing 8048 Instruction Sequences with Single 8x51 Instructions (Continued)

8048 Instruction		Bytes	Cycles	μSec	8x51 Instruction		Bytes	Cycles & μSec
Flag Testing:								
Jump if Software Flag is 0								
JF0	$+4							
JMP	offset	= 4	4	10.0	JNB	F0.rel	3	2
Jump if Accumulator bit is 0								
CPL	A							
JB7	offset							
CPL	A	= 4	4	10.0	JNB	ACC.7.rel	3	2
Peripheral Polling								
Test if Input Pin is Grounded								
IN	A.P1							
CPL	A							
JB3	offset	= 4	5	12.5	JNB	P1.3.rel	3	2
Test if Interrupt Pin is High								
JN1	$+4							
JMP	offset	= 4	4	10.0	JB	INT0.rel	3	2

3.0 BOOLEAN PROCESSOR APPLICATIONS

So what? Then what does all this buy you?

Qualitatively, nothing. All the same capabilities *could* be (and often have been) implemented on other machines using awkward sequences of other basic operations. As mentioned earlier, any CPU can solve any problem given enough time.

Quantitatively, the differences between a solution allowed by the 8051 and those required by previous architectures are numerous. What the 8051 Family buys you is a faster, cleaner, lower-cost solution to microcontroller applications.

The opcode space freed by condensing many specific 8048 instructions into a few general operations has been used to add new functionality to the MCS-51 architecture—both for byte and bit operations. 144 software flags replace the 8048's two. These flags (and the carry) may be directly set, not just cleared and complemented, and all can be tested for either state, not just one. Operating mode bits previously inaccessible may be read, tested, or saved. Situations where the 8051 instruction set provides new capabilities are contrasted with 8048 instruction sequences in Table 4b. Here the 8051 speed advantage ranges from 5x to 15x!

Combining Boolean and byte-wide instructions can produce great synergy. An MCS-51 based application will prove to be:

- simpler to write since the architecture correlates more closely with the problems being solved:
- easier to debug because more individual instructions have no unexpected or undesirable side-effects:
- more byte efficient due to direct bit addressing and program counter relative branching:
- faster running because fewer bytes of instruction need to be fetched and fewer conditional jumps are processed:
- lower cost because of the high level of system-integration within one component.

These rather unabashed claims of excellence shall not go unsubstantiated. The rest of this chapter examines less trivial tasks simplified by the Boolean processor. The first three compare the 8051 with other microprocessors; the last two go into 8051-based system designs in much greater depth.

Design Example # 1—Bit Permutation

First off, we'll use the bit-transfer instructions to permute a lengthy pattern of bits.

A steadily increasing number of data communication products use encoding methods to protect the security of sensitive information. By law, interstate financial transactions involving the Federal banking system must be transmitted using the Federal Information Processing *Data Encryption Standard* (DES).

Basically, the DES combines eight bytes of "plaintext" data (in binary, ASCII, or any other format) with a 56-bit "key", producing a 64-bit encrypted value for transmission. At the receiving end the same algorithm is applied to the incoming data using the same key, reproducing the original eight byte message. The algorithm used for these permutations is fixed; different user-defined keys ensure data privacy.

It is not the purpose of this note to describe the DES in any detail. Suffice it to say that encryption/decryption is a long, iterative process consisting of rotations, exclusive -OR operations, function table look-ups, and an extensive (and quite bizarre) sequence of bit permutation, packing, and unpacking steps. (For further details refer to the June 21, 1979 issue of **Electronics** magazine.) The bit manipulation steps are included, it is rumored, to impede a general purpose digital supercomputer trying to "break" the code. Any algorithm implementing the DES with previous generation microprocessors would spend virtually all of its time diddling bits.

The bit manipulation performed is typified by the Key Schedule Calculation represented in Figure 9. This step is repeated 16 times for each key used in the course of a transmission. In essence, a seven-byte, 56-bit "Shifted Key Buffer" is transformed into an eight-byte, "Permutation Buffer" without altering the shifted Key. The arrows in Figure 9 indicate a few of the translation steps. Only six bits of each byte of the Permutation Buffer are used; the two high-order bits of each byte are cleared. This means only 48 of the 56 Shifted Key Buffer bits are used in any one iteration.

Different microprocessor architectures would best implement this type of permutation in different ways. Most approaches would share the steps of Figure 10a:

- Initialize the Permutation Buffer to default state (ones or zeroes):
- Isolate the state of a bit of a byte from the Key Buffer. Depending on the CPU, this might be accomplished by rotating a word of the Key Buffer through a carry flag or testing a bit in memory or an accumulator against a mask byte:
- Perform a conditional jump based on the carry or zero flag if the Permutation Buffer default state is correct:
- Otherwise reverse the corresponding bit in the permutation buffer with logical operations and mask bytes.

Each step above may require several instructions. The last three steps must be repeated for all 48 bits. Most microprocessors would spend 300 to 3,000 microseconds on each of the 16 iterations.

Notice, though, that this flow chart looks a lot like Figure 8. The Boolean Processor can permute bits by simply moving them from the source to the carry to the destination—a total of two instructions taking four bytes and three microseconds per bit. Assume the Shifted Key Buffer and Permutation Buffer both reside in bit-addressable RAM, with the bits of the former assigned symbolic names SKB__1, SKB__2, . . . SKB__56, and that the bytes of the latter are named PB__1, . . . PB__8. Then working from Figure 9, the software for the permutation algorithm would be that of Example 1a. The total routine length would be 192 bytes, requiring 144 microseconds.

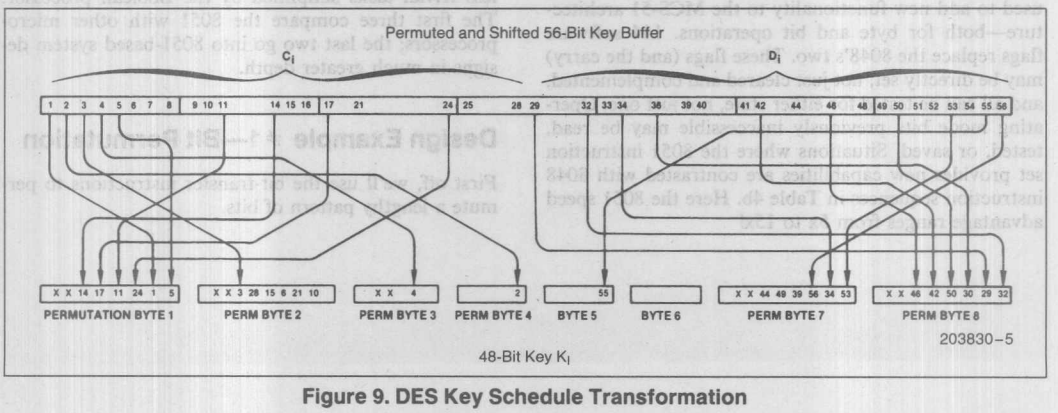


Figure 9. DES Key Schedule Transformation

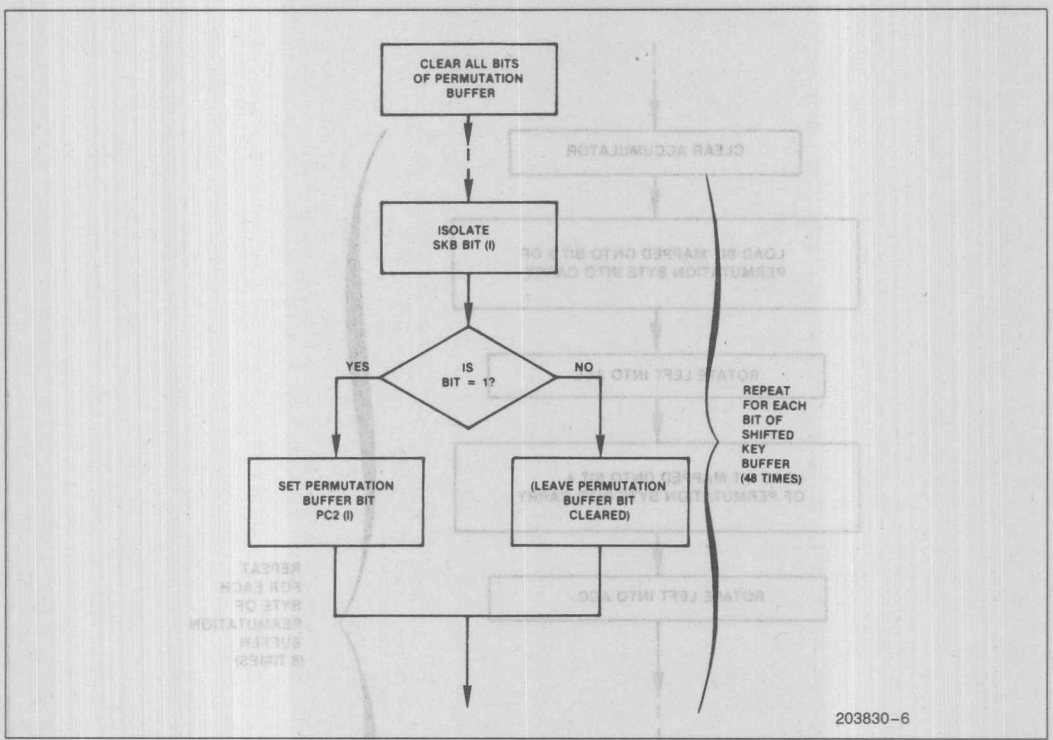


Figure 10a. Flowchart for Key Permutation Attempted with a Byte Processor

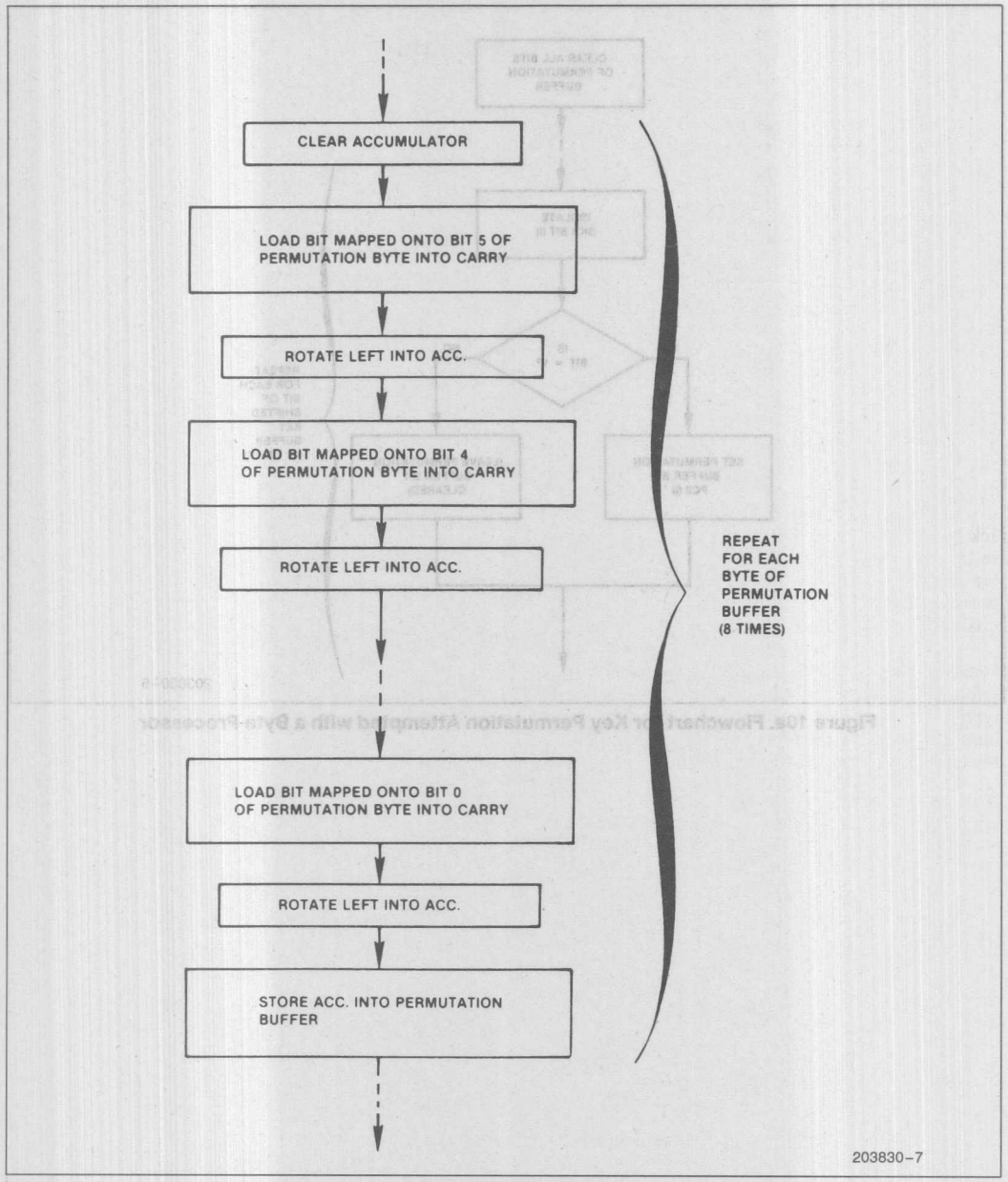


Figure 10b. DES Key Permutation with Boolean Processor

The algorithm of Figure 10b is just slightly more efficient in this time-critical application and illustrates the synergy of an integrated byte and bit processor. The bits needed for each byte of the Permutation Buffer are assimilated by loading each bit into the carry (1 μs.) and shifting it into the accumulator (1 μs.). Each byte is stored in RAM when completed. Forty-eight bits thus need a total of 112 instructions, some of which are listed in Example 1b.

Worst-case execution time would be 112 microseconds, since each instruction takes a single cycle. Routine length would also decrease, to 168 bytes. (Actually, in the context of the complete encryption algorithm, each permuted byte would be processed as soon as it is assimilated—saving memory and cutting execution time by another 8 μs.)

To date, most banking terminals and other systems using the DES have needed special boards or peripheral controller chips just for the encryption/decryption process, and still more hardware to form a serial bit stream for transmission (Figure 11a). An 8051 solution could pack most of the entire system onto the one chip (Figure 11b). The whole DES algorithm would require less than one-fourth of the on-chip program memory, with the remaining bytes free for operating the banking terminal (or whatever) itself.

Moreover, since transmission and reception of data is performed through the on-board UART, the unencrypted data (plaintext) never even exists outside the microcomputer! Naturally, this would afford a high degree of security from data interception.

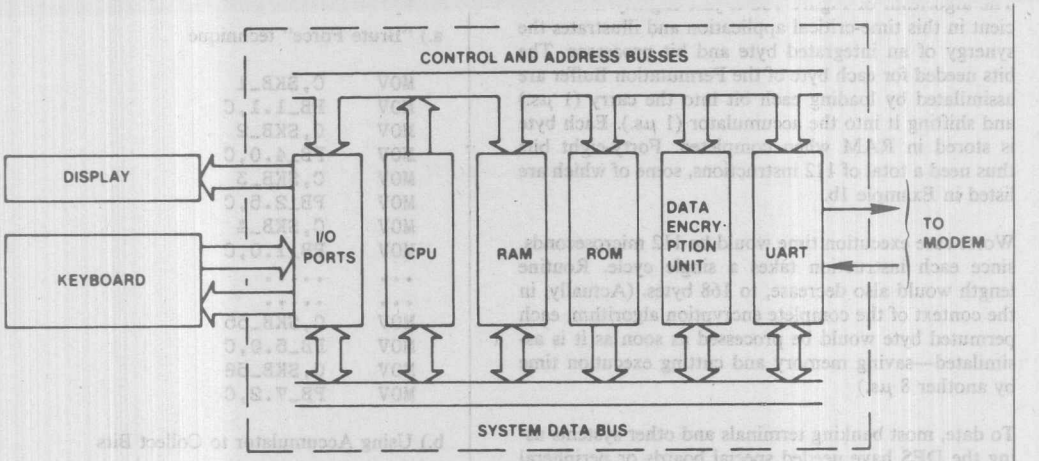
Example 1. DES Key Permutation Software.

a.) "Brute Force" technique

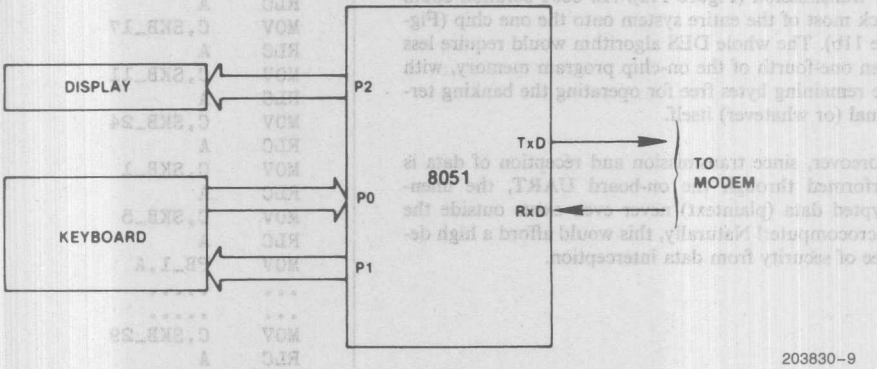
```
MOV     C,SKB_1
MOV     PB_1.1,C
MOV     C,SKB_2
MOV     PB_4.0,C
MOV     C,SKB_3
MOV     PB_2.5,C
MOV     C,SKB_4
MOV     PB_1.0,C
...     .....
...     .....
MOV     C,SKB_55
MOV     PB_5.0,C
MOV     C,SKB_56
MOV     PB_7.2,C
```

b.) Using Accumulator to Collect Bits

```
CLR     A
MOV     C,SKB_14
RLC     A
MOV     C,SKB_17
RLC     A
MOV     C,SKB_11
RLC     A
MOV     C,SKB_24
RLC     A
MOV     C,SKB_1
RLC     A
MOV     C,SKB_5
RLC     A
MOV     PB_1,A
...     .....
...     .....
MOV     C,SKB_29
RLC     A
MOV     C,SKB_32
RLC     A
MOV     PB_8,A
```



a.) Using Multi-Chip Processor Technology



b.) Using One Single-Chip Microcomputer

Figure 11. Secure Banking Terminal Block Diagram

Design Example #2—Software Serial I/O

An exercise often imposed on beginning microcomputer students is to write a program simulating a UART. Though doing this with the 8051 Family may appear to be a moot point (given that the hardware for a full UART is on-chip), it is still instructive to see how it would be done, and maintains a product line tradition.

As it turns out, the 8051 microcomputers can receive or transmit serial data via software very efficiently using the Boolean instruction set. Since any I/O pin may be a serial input or output, several serial links could be maintained at once.

Figures 12a and 12b show algorithms for receiving or transmitting a byte of data. (Another section of program would invoke this algorithm eight times, synchronizing it with a start bit, clock signal, software delay, or timer interrupt.) Data is received by testing an input pin, setting the carry to the same state, shifting the carry into a data buffer, and saving the partial frame in internal RAM. Data is transmitted by shifting an output buffer through the carry, and generating each bit on an output pin.

A side-by-side comparison of the software for this common "bit-banging" application with three different microprocessor architectures is shown in Table 5a and 5b. The 8051 solution is more efficient than the others on every count!

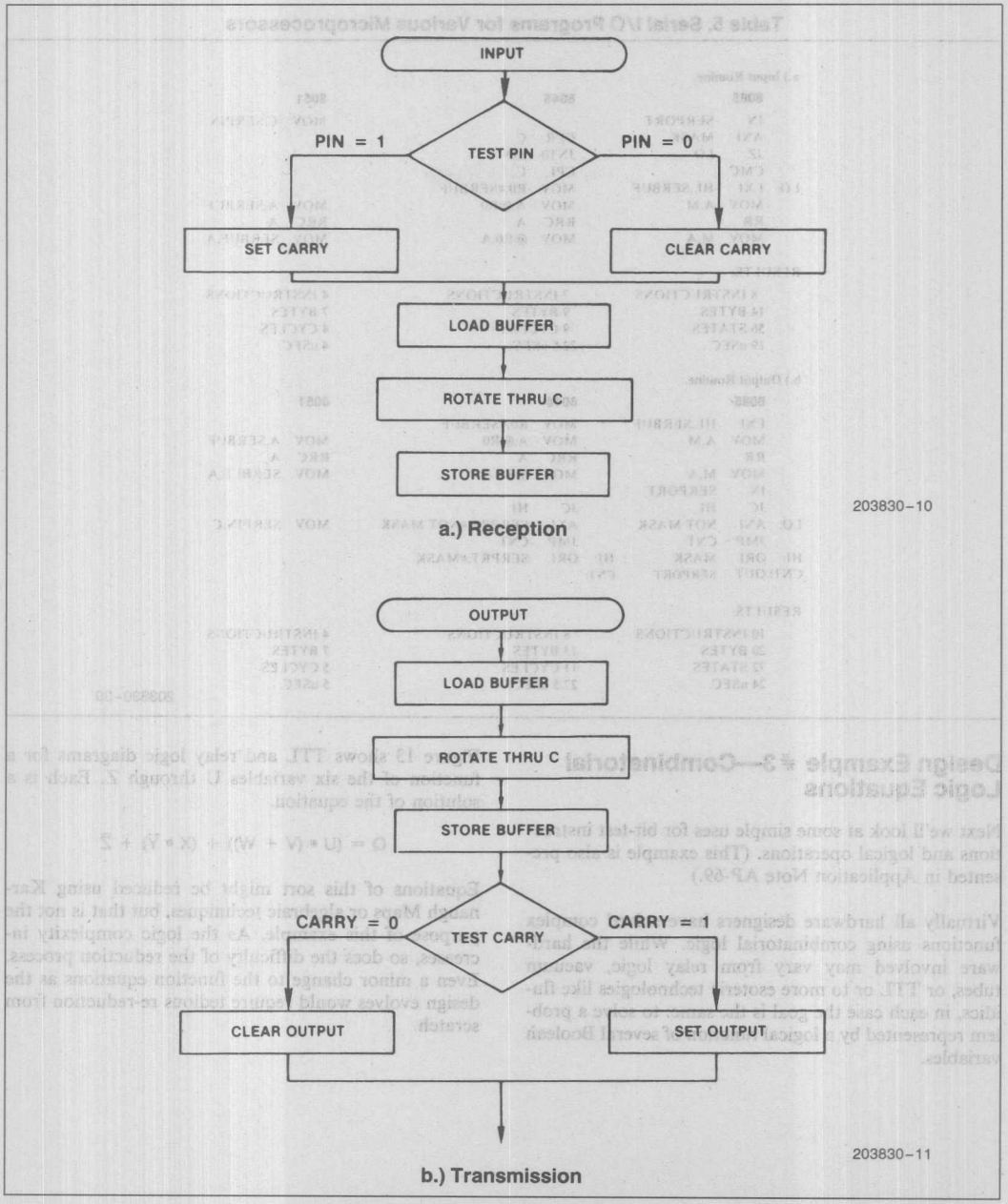


Figure 12. Serial I/O Algorithms

Table 5. Serial I/O Programs for Various Microprocessors

a.) Input Routine.

```
          8085                    8048                    8051
          IN    SERPORT                                   MOV   C,SERPIN
          ANI   MASK              CLR   C
          JZ    LO                JNT0  LO
          CMC                     CPL   C
LO:       LXI   HL,SERBUF         MOV   R0,#SERBUF
          MOV   A,M               MOV   A,@R0             MOV   A,SERBUF
          RR                      RRC   A                 RRC   A
          MOV   M,A               MOV   @R0,A             MOV   SERBUF,A

RESULTS:
          8 INSTRUCTIONS          7 INSTRUCTIONS          4 INSTRUCTIONS
          14 BYTES                9 BYTES                 7 BYTES
          56 STATES               9 CYCLES                4 CYCLES
          19 uSEC.                22.5 uSEC.              4 uSEC.
```

b.) Output Routine.

```
          8085                    8048                    8051
          LXI   HL,SERBUF         MOV   R0,#SERBUF
          MOV   A,M               MOV   A,@R0             MOV   A,SERBUF
          RR                      RRC   A                 RRC   A
          MOV   M,A               MOV   @R0,A             MOV   SERBUF,A
          IN    SERPORT
          JC    HI                JC    HI
LO:       ANI   NOT MASK          ANL   SERPRT,#NOT MASK  MOV   SERPIN,C
          JMP   CNT               JMP   CNT
HI:       ORI   MASK        HI:   ORL   SERPRT,#MASK
CNT:      OUT   SERPORT     CNT:

RESULTS:
          10 INSTRUCTIONS         8 INSTRUCTIONS          4 INSTRUCTIONS
          20 BYTES                13 BYTES                7 BYTES
          72 STATES               11 CYCLES               5 CYCLES
          24 uSEC.                27.5 uSEC.              5 uSEC.
```

203830-30

Design Example #3—Combinatorial Logic Equations

Next we'll look at some simple uses for bit-test instructions and logical operations. (This example is also presented in Application Note AP-69.)

Virtually all hardware designers have solved complex functions using combinatorial logic. While the hardware involved may vary from relay logic, vacuum tubes, or TTL or to more esoteric technologies like fluidics, in each case the goal is the same: to solve a problem represented by a logical function of several Boolean variables.

Figure 13 shows TTL and relay logic diagrams for a function of the six variables U through Z. Each is a solution of the equation.

$$Q = (U \bullet (V + W)) + (X \bullet \overline{Y}) + \overline{Z}$$

Equations of this sort might be reduced using Karnaugh Maps or algebraic techniques, but that is not the purpose of this example. As the logic complexity increases, so does the difficulty of the reduction process. Even a minor change to the function equations as the design evolves would require tedious re-reduction from scratch.

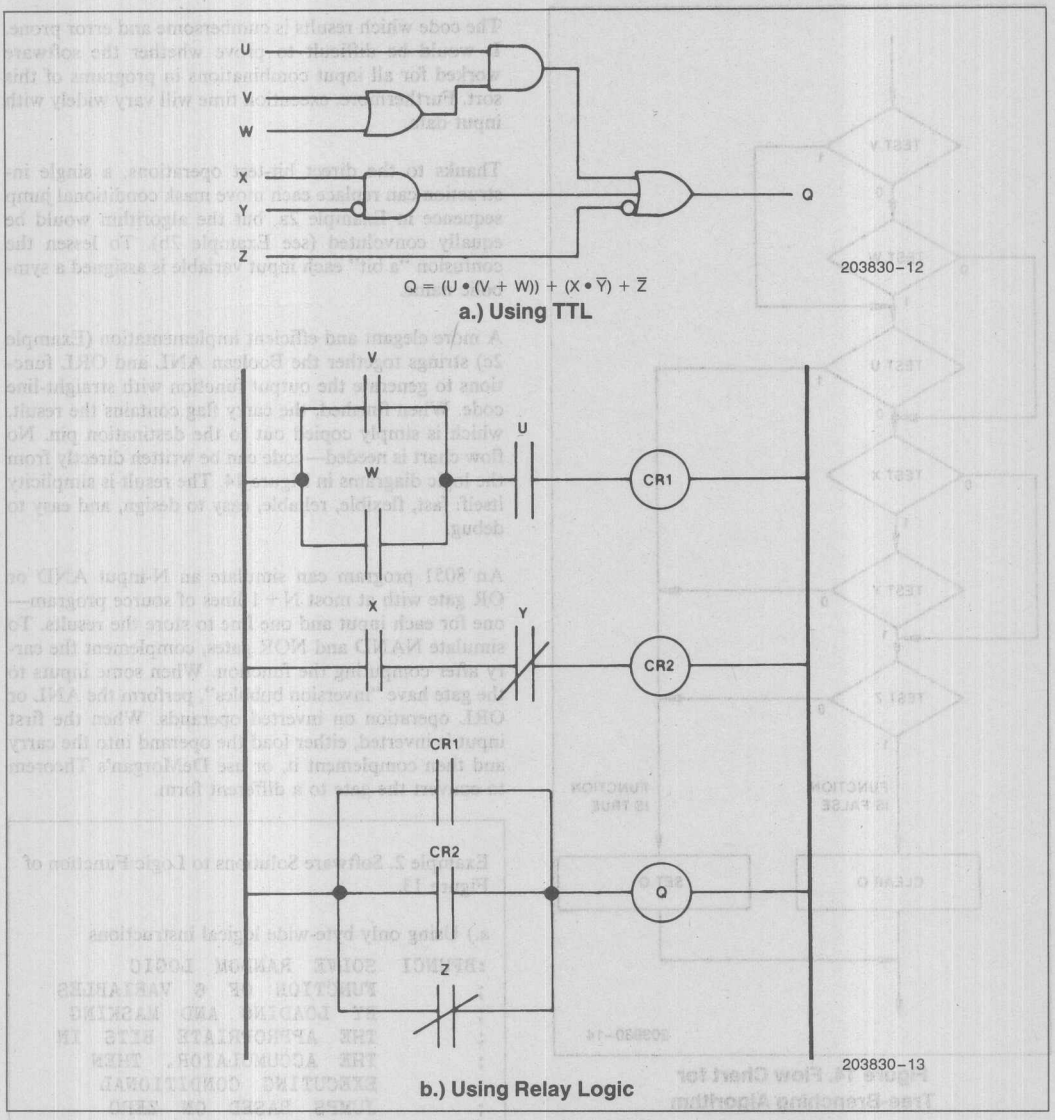


Figure 13. Hardware Implementations of Boolean Functions

For the sake of comparison we will implement this function three ways, restricting the software to three proper subsets of the MCS-51 instruction set. We will also assume that U and V are input pins from different input ports, W and X are status bits for two peripheral controllers, and Y and Z are software flags set up earlier in the program. The end result must be written to an output pin on some third port. The first two implementations follow the flow-chart shown in Figure 14. Program flow would embark on a route down a test-and-branch tree and leaves either the "True" or "Not True" exit ASAP—as soon as the proper result has been determined. These exits then rewrite the output port with the result bit respectively one or zero.

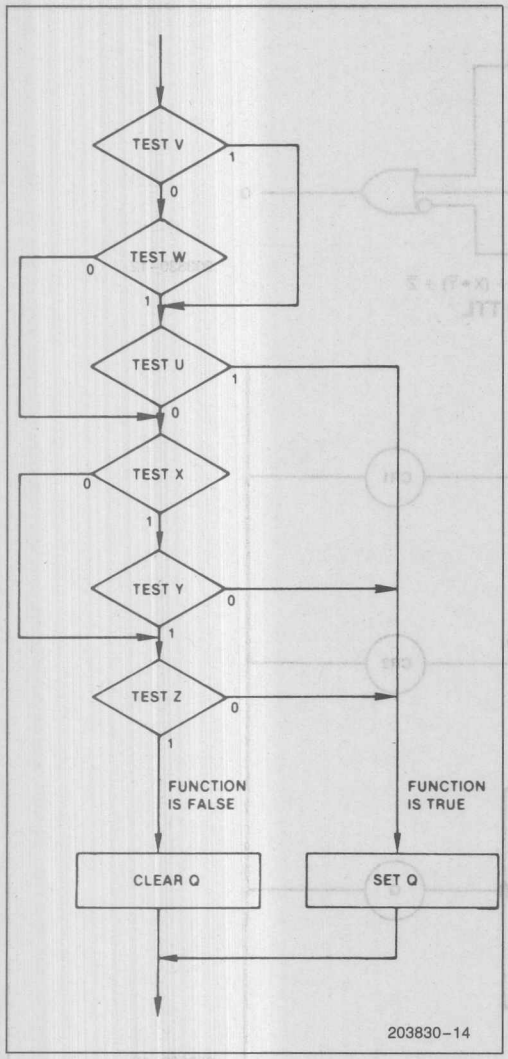


Figure 14. Flow Chart for Tree-Branching Algorithm

Other digital computers must solve equations of this type with standard word-wide logical instructions and conditional jumps. So for the first implementation, we won't use any generalized bit-addressing instructions. As we shall soon see, being constrained to such an instruction subset produces somewhat sloppy software solutions. MCS-51 mnemonics are used in Example 2a: other machines might further cloud the situation by requiring operation-specific mnemonics like INPUT, OUTPUT, LOAD, STORE, etc., instead of the MOV mnemonic used for all variable transfers in the 8051 instruction set.

The code which results is cumbersome and error prone. It would be difficult to prove whether the software worked for all input combinations in programs of this sort. Furthermore, execution time will vary widely with input data.

Thanks to the direct bit-test operations, a single instruction can replace each move mask conditional jump sequence in Example 2a, but the algorithm would be equally convoluted (see Example 2b). To lessen the confusion "a bit" each input variable is assigned a symbolic name.

A more elegant and efficient implementation (Example 2c) strings together the Boolean ANL and ORL functions to generate the output function with straight-line code. When finished, the carry flag contains the result, which is simply copied out to the destination pin. No flow chart is needed—code can be written directly from the logic diagrams in Figure 14. The result is simplicity itself: fast, flexible, reliable, easy to design, and easy to debug.

An 8051 program can simulate an N-input AND or OR gate with at most N + 1 lines of source program—one for each input and one line to store the results. To simulate NAND and NOR gates, complement the carry after computing the function. When some inputs to the gate have "inversion bubbles", perform the ANL or ORL operation on inverted operands. When the first input is inverted, either load the operand into the carry and then complement it, or use DeMorgan's Theorem to convert the gate to a different form.

Example 2. Software Solutions to Logic Function of Figure 13.

a.) Using only byte-wide logical instructions

```
:BFUNCI  SOLVE RANDOM LOGIC
;        FUNCTION OF 6 VARIABLES
;        BY LOADING AND MASKING
;        THE APPROPRIATE BITS IN
;        THE ACCUMULATOR. THEN
;        EXECUTING CONDITIONAL
;        JUMPS BASED ON ZERO
;        CONDITION. (APPROACH USED
;        BY BYTE-ORIENTED
;        ARCHITECTURES.) BYTE AND
;        MASK VALUES CORRESPOND TO
;        RESPECTIVE BYTE ADDRESS
;        AND BIT POSITIONS.
;
OUTBUF   DATA 22H
;OUTPUT PIN STATE MAP
;
```

```
TESTV:  MOV   A,P2
        ANL   A,#00000100B
        JNZ   TESTU
        MOV   A,TCON
        ANL   A,#00100000B
        JZ    TESTX
TESTU:  MOV   A,P1
        ANL   A,#00000010B
        JNZ   SETQ
TESTX:  MOV   A,TCON
        ANL   A,#00001000B
        JZ    TESTZ
        MOV   A,20H
        ANL   A,#00000001B
        JZ    SETQ
TESTZ:  MOV   A,21H
        ANL   A,#00000010B
        JZ    SETQ
CLRQ:   MOV   A,OUTBUF
        ANL   A,#11110111B
        JMP   OUTQ
SETQ:   MOV   A,OUTBUF
        ORL   A,#00001000B
OUTQ:   MOV   OUTBUF,A
        MOV   P3,A
```

b.) Using only bit-test instructions

```
:BFUNC2  SOLVE A RANDOM LOGIC
;        FUNCTION OF 6 VARIABLES
;        BY DIRECTLY POLLING EACH
;        BIT. (APPROACH USING
;        MCS-51 UNIQUE BIT-TEST
;        INSTRUCTION CAPABILITY.)
;        SYMBOLS USED IN LOGIC
;        DIAGRAM ASSIGNED TO
;        CORRESPONDING 8x51 BIT
;        ADDRESSES.
;
U       BIT   P1.1
V       BIT   P2.2
W       BIT   TF0
X       BIT   IE1
Y       BIT   20H.0
Z       BIT   21H.1
Q       BIT   P3.3
;       ...   ....
TEST_V: JB    V,TEST_U
        JNB   W,TEST_X
TEST_U: JB    U,SET_Q
TEST_X: JNB   X,TEST_Z
        JNB   Y,SET_Q
TEST_Z: JNB   Z,SET_Q
CLR_Q:  CLR   Q
        JMP   NXTTST
SET_Q:  SETB  Q
NXTTST:(CONTINUATION OF
       :PROGRAM)
```

c.) Using logical operations on Boolean variables

```
:FUNC3  SOLVE A RANDOM LOGIC
;       FUNCTION OF 6 VARIABLES
;       USING STRAIGHT_LINE
;       LOGICAL INSTRUCTIONS ON
;       MCS-51 BOOLEAN VARIABLES.
;
MOV C,V
ORL C,W   ;OUTPUT OF OR GATE
ANL C,U   ;OUTPUT OF TOP AND GATE
MOV F0,C  ;SAVE INTERMEDIATE STATE
MOV C,X
ANL C,Y   ;OUTPUT OF BOTTOM AND GATE
ORL C,F0  ;INCLUDE VALUE SAVED ABOVE
ORL C,Z   ;INCLUDE LAST INPUT
          ;VARIABLE
MOV Q,C   ;OUTPUT COMPUTED RESULT
```

An upper-limit can be placed on the complexity of software to simulate a large number of gates by summing the total number of inputs and outputs. The *actual* total should be somewhat shorter, since calculations can be "chained," as shown. The output of one gate is often the first input to another, bypassing the intermediate variable to eliminate two lines of source.

Design Example #4—Automotive Dashboard Functions

Now let's apply these techniques to designing the software for a complete controller system. This application is patterned after a familiar real-world application which isn't nearly as trivial as it might first appear: automobile turn signals.

Imagine the three position turn lever on the steering column as a single-pole, triple-throw toggle switch. In its central position all contacts are open. In the up or down positions contacts close causing corresponding lights in the rear of the car to blink. So far very simple.

Two more turn signals blink in the front of the car, and two others in the dashboard. All six bulbs flash when an emergency switch is closed. A thermo-mechanical relay (accessible under the dashboard in case it wears out) causes the blinking.

Applying the brake pedal turns the tail light filaments on constantly . . . unless a turn is in progress, in which case the blinking tail light is not affected. (Of course, the front turn signals and dashboard indicators are not affected by the brake pedal.) Table 6 summarizes these operating modes.

Table 6. Truth Table for Turn-Signal Operation

Input Signals				Output Signals			
Brake Switch	Emerg. Switch	Left Turn Switch	Right Turn Switch	Left Front & Dash	Right Front & Dash	Left Rear	Right Rear
0	0	0	0	Off	Off	Off	Off
0	0	0	1	Off	Blink	Off	Blink
0	0	1	0	Blink	Off	Blink	Off
0	1	0	0	Blink	Blink	Blink	Blink
0	1	0	1	Blink	Blink	Blink	Blink
0	1	1	0	Blink	Blink	Blink	Blink
1	0	0	0	Off	Off	On	On
1	0	0	1	Off	Blink	On	Blink
1	0	1	0	Blink	Off	Blink	On
1	1	0	0	Blink	Blink	On	On
1	1	0	1	Blink	Blink	On	Blink
1	1	1	0	Blink	Blink	Blink	On

But we're not done yet. Each of the exterior turn signal (but not the dashboard) bulbs has a second, somewhat dimmer filament for the parking lights. Figure 15 shows TTL circuitry which could control all six bulbs. The signals labeled "High Freq." and "Low Freq." represent two square-wave inputs. Basically, when one of the turn switches is closed or the emergency switch is activated the low frequency signal (about 1 Hz) is gated through to the appropriate dashboard indicator(s) and turn signal(s). The rear signals are also activated when the brake pedal is depressed provided a turn is not being made in the same direction. When the parking light switch is closed the higher frequency oscillator is gated to each front and rear turn signal, sustaining a low-intensity background level. (This is to eliminate the need for additional parking light filaments.)

In most cars, the switching logic to generate these functions requires a number of multiple-throw contacts. As many as 18 conductors thread the steering column of some automobiles solely for turn-signal and emergency blinker functions. (The author discovered this recently to his astonishment and dismay when replacing the whole assembly because of one burned contact.)

A multiple-conductor wiring harness runs to each corner of the car, behind the dash, up the steering column, and down to the blinker relay below. Connectors at each termination for each filament lead to extra cost and labor during construction, lower reliability and safety, and more costly repairs. And considering the system's present complexity, increasing its reliability or detecting failures would be quite difficult.

There are two reasons for going into such painful detail describing this example. First, to show that the messiest part of many system designs is determining what the controller should do. Writing the software to solve these functions will be comparatively easy. Secondly, to show the many potential failure points in the system. Later we'll see how the peripheral functions and intelligence built into a microcomputer (with a little creativity) can greatly reduce external interconnections and mechanical part count.

The Single-Chip Solution

The circuit shown in Figure 16 indicates five input pins to the five input variables—left-turn select, right-turn select, brake pedal down, emergency switch on, and parking lights on. Six output pins turn on the front, rear, and dashboard indicators for each side. The microcomputer implements all logical functions through software, which periodically updates the output signals as time elapses and input conditions change.

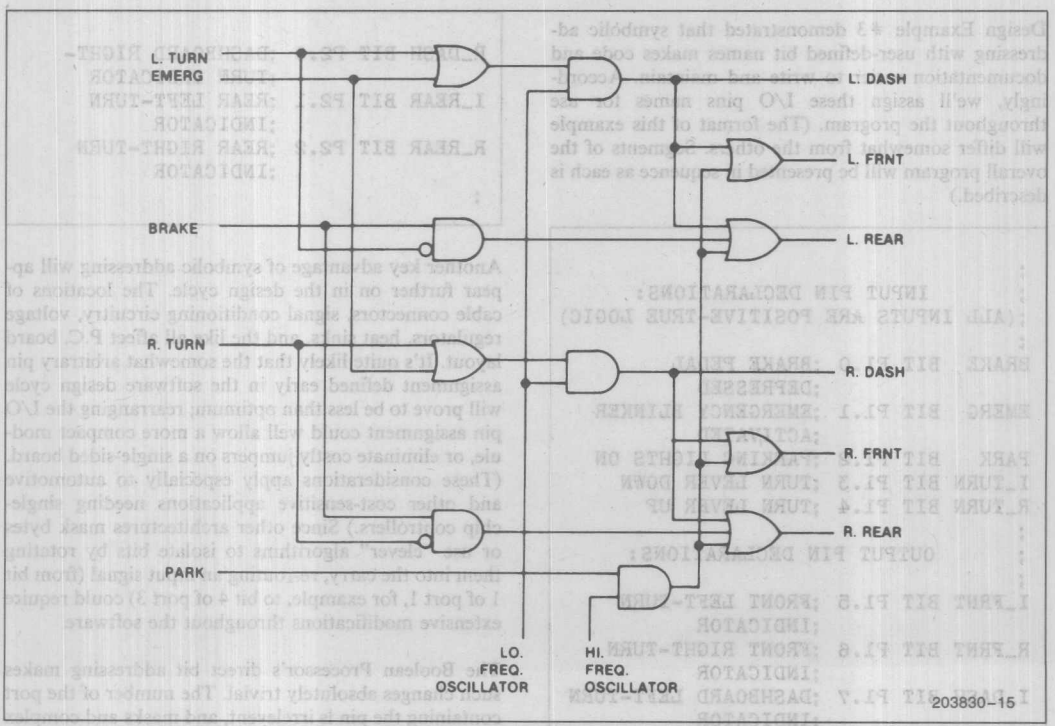


Figure 15. TTL Logic Implementation of Automotive Turn Signals

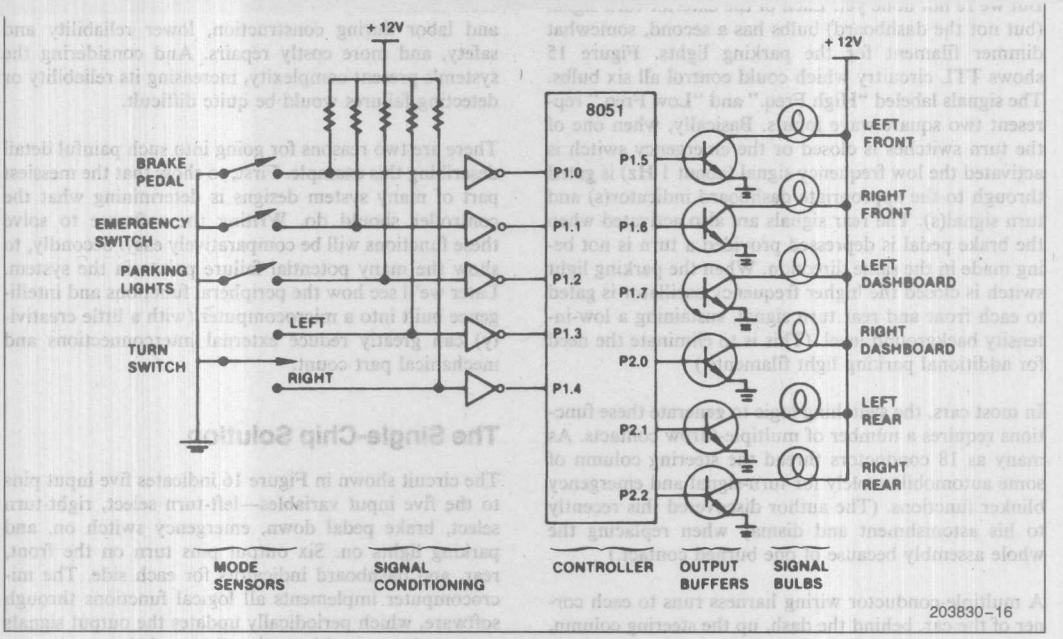


Figure 16. Microcomputer Turn-Signal Connections

Design Example #3 demonstrated that symbolic addressing with user-defined bit names makes code and documentation easier to write and maintain. Accordingly, we'll assign these I/O pins names for use throughout the program. (The format of this example will differ somewhat from the others. Segments of the overall program will be presented in sequence as each is described.)

```
;
;       INPUT PIN DECLARATIONS:
;(ALL INPUTS ARE POSITIVE-TRUE LOGIC)
;
BRAKE   BIT P1.0 ;BRAKE PEDAL
                 ;DEPRESSED
EMERG   BIT P1.1 ;EMERGENCY BLINKER
                 ;ACTIVATED
PARK    BIT P1.2 ;PARKING LIGHTS ON
I_TURN  BIT P1.3 ;TURN LEVER DOWN
R_TURN  BIT P1.4 ;TURN LEVER UP
;
;       OUTPUT PIN DECLARATIONS:
;
I_FRNT  BIT P1.5 ;FRONT LEFT-TURN
                 ;INDICATOR
R_FRNT  BIT P1.6 ;FRONT RIGHT-TURN
                 ;INDICATOR
I_DASH  BIT P1.7 ;DASHBOARD LEFT-TURN
                 ;INDICATOR
```

```
R_DASH  BIT P2.0 ;DASHBOARD RIGHT-
                 ;TURN INDICATOR
I_REAR  BIT P2.1 ;REAR LEFT-TURN
                 ;INDICATOR
R_REAR  BIT P2.2 ;REAR RIGHT-TURN
                 ;INDICATOR
;
```

Another key advantage of symbolic addressing will appear further on in the design cycle. The locations of cable connectors, signal conditioning circuitry, voltage regulators, heat sinks, and the like all affect P.C. board layout. It's quite likely that the somewhat arbitrary pin assignment defined early in the software design cycle will prove to be less than optimum; rearranging the I/O pin assignment could well allow a more compact module, or eliminate costly jumpers on a single-sided board. (These considerations apply especially to automotive and other cost-sensitive applications needing single-chip controllers.) Since other architectures mask bytes or use "clever" algorithms to isolate bits by rotating them into the carry, re-routing an input signal (from bit 1 of port 1, for example, to bit 4 of port 3) could require extensive modifications throughout the software.

The Boolean Processor's direct bit addressing makes such changes absolutely trivial. The number of the port containing the pin is irrelevent, and masks and complex

program structures are not needed. Only the initial Boolean variable declarations need to be changed; ASM51 automatically adjusts all addresses and symbolic references to the reassigned variables. The user is assured that no additional debugging or software verification will be required.

```
;           ...         .....
;INTERRUPT RATE SUBDIVIDER
SUB_DIV   DATA      20H
;HIGH-FREQUENCY OSCILLATOR BIT
HI_FREQ   BIT       SUB_DIV,0
;LOW-FREQUENCY OSCILLATOR BIT
LO_FREQ   BIT       SUB_DIV,7
;           ...
            ORG     0000H
JMP         INIT
;           ...     .....
            ORG     100H
;PUT TIMER 0 IN MODE 1
INIT;       MOV     TMOD,#00000001B
;INITIALIZE TIMER REGISTERS
            MOV     TL0,#0
            MOV     TH0,#-16
;SUBDIVIDE INTERRUPT RATE BY 244
            MOV     SUB_DIV,#244
;ENABLE TIMER INTERRUPTS
            SETB    ET0
;GLOBALLY ENABLE ALL INTERRUPTS
            SETB    EA
;START TIMER
            SETB    TR0
;
;(CONTINUE WITH BACKGROUND PROGRAM)
;
;PUT TIMER 0 IN MODE 1
;INITIALIZE TIMER REGISTERS

;SUBDIVIDE INTERRUPT RATE BY 244
;ENABLE TIMER INTERRUPTS
;GLOBALLY ENABLE ALL INTERRUPTS
;START TIMER
```

Timer 0 (one of the two on-chip timer counters) replaces the thermo-mechanical blinker relay in the dashboard controller. During system initialization it is configured as a timer in mode 1 by setting the least significant bit of the timer mode register (TMOD). In this configuration the low-order byte (TL0) is incremented every machine cycle, overflowing and incrementing the high-order byte (TH0) every 256 μs. Timer interrupt 0 is enabled so that a hardware interrupt will occur each time TH0 overflows.

An eight-bit variable in the bit-addressable RAM array will be needed to further subdivide the interrupts via software. The lowest-order bit of this counter toggles very fast to modulate the parking lights: bit 7 will be "tuned" to approximately 1 Hz for the turn- and emergency-indicator blinking rate.

Loading TH0 with -16 will cause an interrupt after 4.096 ms. The interrupt service routine reloads the high-order byte of timer 0 for the next interval, saves the CPU registers likely to be affected on the stack, and then decrements SUB_DIV. Loading SUB_DIV. with 244 initially and each time it decrements to zero will produce a 0.999 second period for the highest-order bit.

```
ORG   000BH   ;TIMER 0 SERVICE VECTOR
MOV   TH0,#-16
PUSH  PSW
PUSH  ACC
PUSH  B
DJNZ  SUB_DIV,T0SERV
MOV   SUB_DIV,#244
```

The code to sample inputs, perform calculations, and update outputs—the real "meat" of the signal controller algorithm—may be performed either as part of the interrupt service routine or as part of a background program loop. The only concern is that it must be executed at least serveral dozen times per second to prevent parking light flickering. We will assume the former case, and insert the code into the timer 0 service routine.

First, notice from the logic diagram (Figure 15) that the subterm (PARK • H_FREQ), asserted when the parking lights are to be on dimly, figures into four of the six output functions. Accordingly, we will first compute that term and save it in a temporary location named "DIM". The PSW contains two general purpose flags: F0, which corresponds to the 8048 flag of the same name, and PSW.1. Since the PSW has been saved and will be restored to its previous state after servicing the interrupt, we can use either bit for temporary storage.

```
  DIM BIT   PSW.1 ;DECLARE TEMP
                  ;STORAGE FLAG
; ... .....
  MOV C,PARK      ;GATE PARKING
                  ;LIGHT SWITCH
  ANL HI_FREQ     ;WITH HIGH
                  ;FREQUENCY
                  ;SIGNAL
  MOV DIM,C       ;AND SAVE IN
                  ;TEMP. VARIABLE
```

This simple three-line section of code illustrates a remarkable point. The software indicates in very abstract terms exactly what function is being performed, inde-

pendent of the hardware configuration. The fact that these three bits include an input pin, a bit within a program variable, and a software flag in the PSW is totally invisible to the programmer.

Now generate and output the dashboard left turn signal.

```
;
   MOV C,L_TURN      ;SET CARRY IF
                     ;TURN
   ORL C,EMERG       ;OR EMERGENCY
                     ;SELECTED
   ANL C,LO_FREQ     ;GATE IN 1 HZ
                     ;SIGNAL
   MOV I_DASH,C      ;AND OUTPUT TO
                     ;DASHBOARD
```

To generate the left front turn signal we only need to add the parking light function in F0. But notice that the function in the carry will also be needed for the rear signal. We can save effort later by saving its current state in F0.

```
;
   MOV F0,C          ;SAVE FUNCTION
                     ;SO FAR
   ORL C,DIM         ;ADD IN PARKING
                     ;LIGHT FUNCTION
   MOV L_FRNT,C      ;AND OUTPUT TO
                     ;TURN SIGNAL
```

Finally, the rear left turn signal should also be on when the brake pedal is depressed, provided a left turn is not in progress.

```
MOV C,BRAKE          ;GATE BRAKE
                     ;PEDAL SWITCH
ANL C,L_TURN         ;WITH TURN
                     ;LEVER
ORL C,FO             ;INCLUDE TEMP.
                     ;VARIABLE FROM DASH
ORL C,DIM            ;AND PARKING
                     ;LIGHT FUNCTION
MOV L_REAR,C         ;AND OUTPUT TO
                     ;TURN SIGNAL
```

Now we have to go through a similar sequence for the right-hand equivalents to all the left-turn lights. This also gives us a chance to see how the code segments above look when combined.

```
MOV C.R_TURN      ;SET CARRY H-
                  ;TURN
ORL C.EMERG       ;OR EMERGENCY
                  ;SELECTED
ANL C,LO_FREQ     ;IF SO. GATE IN 1
                  ;HZ SIGNAL
MOV R_DASH.C      ;AND OUTPUT TO
                  ;DASHBOARD
MOV FO.C          ;SAVE FUNCTION
                  ;SO FAR
ORL C.DIM         ;ADD IN PARKING
                  ;LIGHT FUNCTION
MOV R_FRNT.C      ;AND OUTPUT TO
                  ;TURN SIGNAL
MOV C.BRAKE       ;GATE BRAKE
                  ;PEDAL SWITCH
ANL C. R_TURN     ;WITH TURN
                  ;LEVER
ORL C.FO          ;INCLUDE TEMP.
                  ;VARIABLE FROM
                  ;DASH
ORL C.DIM         ;AND PARKING
                  ;LIGHT FUNCTION
MOV R_REAR.C      ;AND OUTPUT TO
                  ;TURN SIGNAL
```

(The perceptive reader may notice that simply rearranging the steps could eliminate one instruction from each sequence.)

Now that all six bulbs are in the proper states, we can return from the interrupt routine, and the program is finished. This code essentially needs to reverse the status saving steps at the beginning of the interrupt.

Table 7. Non-Trivial Duty Cycles

Sub_Div Bits								Duty Cycles						
7	6	5	4	3	2	1	0	12.5%	25.0%	37.5%	50.0%	62.5%	75.0%	87.5%
X	X	X	X	X	0	0	0	Off	Off	Off	Off	Off	Off	Off
X	X	X	X	X	0	0	1	Off	Off	Off	Off	Off	Off	On
X	X	X	X	X	0	1	0	Off	Off	Off	Off	Off	On	On
X	X	X	X	X	0	1	1	Off	Off	Off	Off	On	On	On
X	X	X	X	X	1	0	0	Off	Off	Off	On	On	On	On
X	X	X	X	X	1	0	1	Off	Off	On	On	On	On	On
X	X	X	X	X	1	1	0	Off	On	On	On	On	On	On
X	X	X	X	X	1	1	1	On	On	On	On	On	On	On

```
POP  B          ;RESTORE CPU
                ;REGISTERS.
POP ACC
POP PSW
RETI
```

Program Refinements. The luminescence of an incandescent light bulb filament is generally non-linear: the 50% duty cycle of HI_FREQ may not produce the desired intensity. If the application requires, duty cycles of 25%, 75%, etc. are easily achieved by ANDing and ORing in additional low-order bits of SUB_DIV. For example, 30 H/ signals of seven different duty cycles could be produced by considering bits 2–0 as shown in Table 7. The only software change required would be to the code which sets-up variable DIM;

```
MOV C,SUB_DIV.1 ;START WITH 50
                ;PERCENT
ANL C,SUB_DIV.0 ;MASK DOWN TO 25
                ;PERCENT
ORL C,SUB_DIV.2 ;AND BUILD BACK TO
                ;62 PERCENT
MOV DIM,C       ;DUTY CYCLE FOR
                ;PARKING LIGHTS.
```

Interconnections increase cost and decrease reliability. The simple buffered pin-per-function circuit in Figure 16 is insufficient when many outputs require higher-than-TTL drive levels. A lower-cost solution uses the 8051 serial port in the shift-register mode to augment I/O. In mode 0, writing a byte to the serial port data buffer (SBUF) causes the data to be output sequentially through the "RXD" pin while a burst of eight clock pulses is generated on the "TXD" pin. A shift register connected to these pins (Figure 17) will load the data byte as it is shifted out. A number of special peripheral driver circuits combining shift-register inputs with high drive level outputs have been introduced recently.

Cascading multiple shift registers end-to-end will expand the number of outputs even further. The data rate in the I/O expansion mode is one megabaud, or 8 μs. per byte. This is the mode which the serial port defaults to following a reset, so no initialization is required.

The software for this technique uses the B register as a "map" corresponding to the different output functions. The program manipulates these bits instead of the output pins. After all functions have been calculated the B register is shifted by the serial port to the shift-register driver. (While some outputs may glitch as data is shifted through them, at 1 Megabaud most people wouldn't notice. Some shift registers provide an "enable" bit to hold the output states while new data is being shifted in.)

This is where the earlier decision to address bits symbolically throughout the program is going to pay off. This major I/O restructuring is nearly as simple to implement as rearranging the input pins. Again, only the bit declarations need to be changed.

```
I_FRNT BIT B.0  ;FRONT LEFT-TURN
                ;INDICATOR
R_FRNT BIT B.1  ;FRONT RIGHT-TURN
                ;INDICATOR
I_DASH BIT B.2  ;DASHBOARD LEFT-TURN
                ;INDICATOR
R_DASH BIT B.3  ;DASHBOARD RIGHT-TURN
                ;INDICATOR
I_REAR BIT B.4  ;REAR LEFT-TURN
                ;INDICATOR
R_REAR BIT B.5  ;REAR RIGHT-TURN
                ;INDICATOR
```

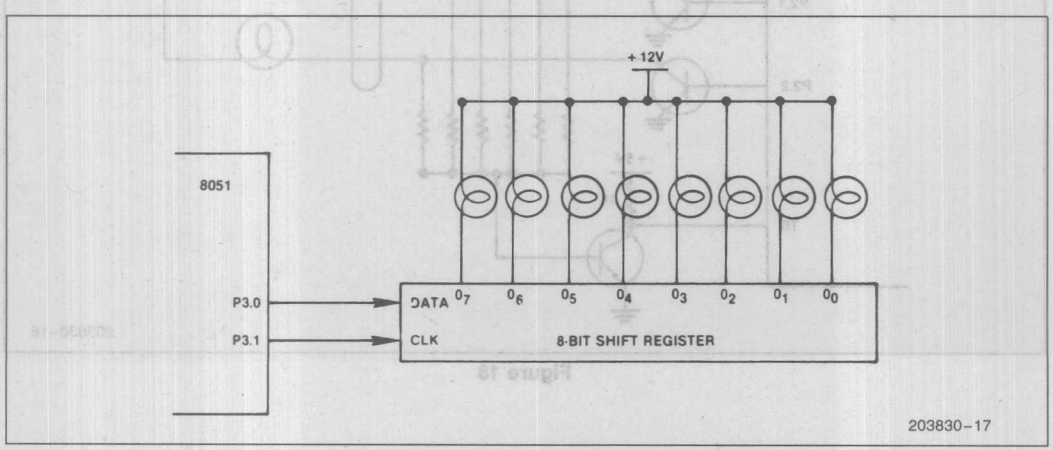


Figure 17. Output Expansion Using Serial Port

The original program to compute the functions need not change. After computing the output variables, the control map is transmitted to the buffered shift register through the serial port.

```
MOV SBUF,B ;LOAD BUFFER AND TRANSMIT
```

The Boolean Processor solution holds a number of advantages over older methods. Fewer switches are required. Each is simpler, requiring fewer poles and lower current contacts. The flasher relay is eliminated entirely. Only six filaments are driven, rather than 10. The wiring harness is therefore simpler and less expensive—one conductor for each of the six lamps and each of the five sensor switches. The fewer conductors use far fewer connectors. The whole system is more reliable.

And since the system is much simpler it would be feasible to implement redundancy and or fault detection on the four main turn indicators. Each could still be a standard double filament bulb, but with the filaments driven in parallel to tolerate single-element failures.

Even with redundancy, the lights will eventually fail. To handle this inescapable fact current or voltage sensing circuits on each main drive wire can verify that each bulb and its high-current driver is functioning properly. Figure 18 shows one such circuit.

Assume all of the lights are turned on except one: i.e., all but one of the collectors are grounded. For the bulb which is turned off, if there is continuity from +12V through the bulb base and filament, the control wire, all connectors, and the P.C. board traces, and if the transistor is indeed not shorted to ground, then the collector will be pulled to +12V. This turns on the base of Q8 through the corresponding resistor, and grounds the input pin, verifying that the bulb circuit is operational. The continuity of each circuit can be checked by software in this way.

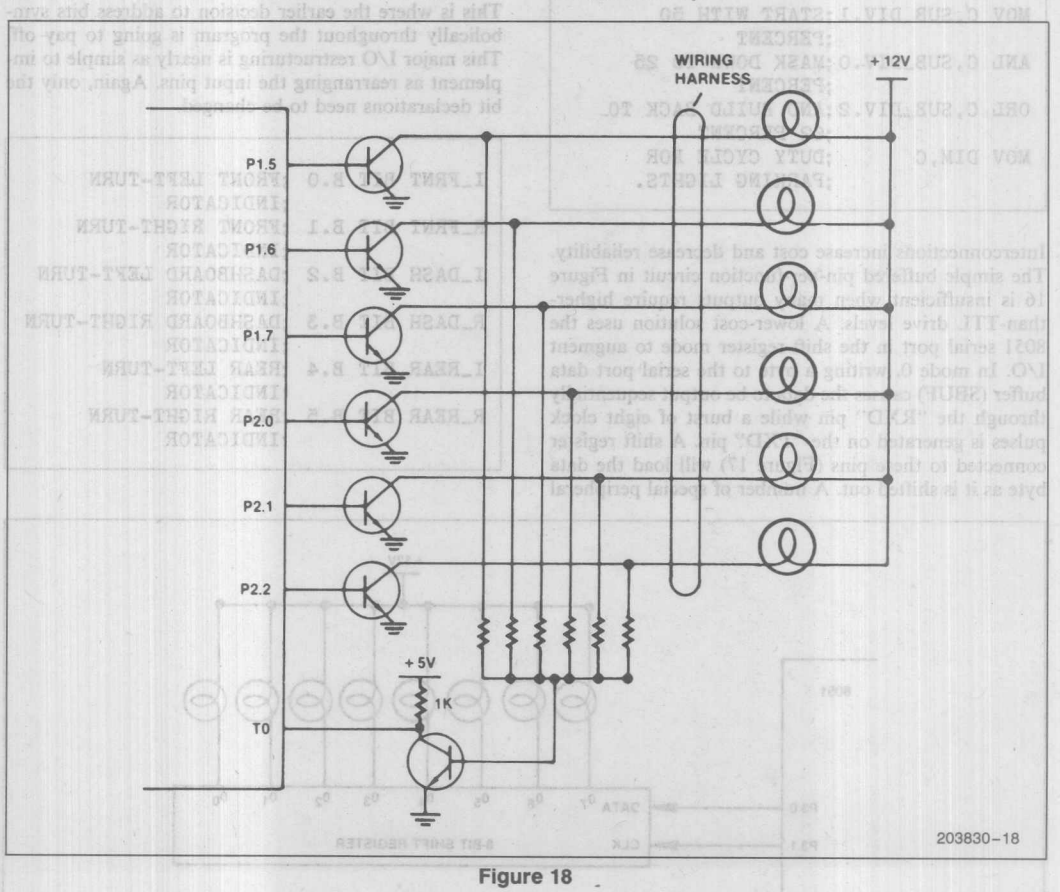


Figure 18

Now turn *all* the bulbs on, grounding all the collectors. Q7 should be turned off, and the Test pin should be high. However, a control wire shorted to +12V or an open-circuited drive transistor would leave one of the collectors at the higher voltage even now. This too would turn on Q7, indicating a different type of failure. Software could perform these checks once per second by executing the routine every time the software counter SUB_DIV is reloaded by the interrupt routine.

```
DJNZ SUB_DIV,TOSERV
MOV SUB_DIV,#244      ;RELOAD COUNTER
ORL P1,#11100000B     ;SET CONTROL
                      ;OUTPUTS HIGH
ORL P2,#00000111B
CLR I_FRNT            ;FLOAT DRIVE
                      ;COLLECTOR
JB  TO,FAULT          ;TO SHOULD BE
                      ;PULLED LOW
SETB L_FRNT           ;PULL COLLECTOR
                      ;BACK DOWN
    CLR  L_DASH
    JB   TO,FAULT
    SETB L_DASH
    CLR  L_REAR
    JB   TO,FAULT
    SETB L_REAR
    CLR  R_FRNT
    JB   TO,FAULT
    SETB R_FRNT
    CLR  R_DASH
    JB   TO,FAULT
    SETB R_DASH
    CLR  R_REAR
    JB   TO,FAULT
    SETB R_REAR
;
;WITH ALL COLLECTORS GROUNDED. TO
 SHOULD BE HIGH
;IF SO. CONTINUE WITH INTERRUPT
 ROUTINE.
  JB   TO,TOSERV
FAULT:                ;ELECTRICAL
                      ;FAILURE
                      ;PROCESSING
                      ;ROUTINE
                      ;(LEFT TO
                      ;READER'S
                      ;IMAGINATION)
TOSERV:               ;CONTINUE WITH
                      ;INTERRUPT
                      ;PROCESSING
;
;
```

The complete assembled program listing is printed in Appendix A. The resulting code consists of 67 program statements, not counting declarations and comments, which assemble into 150 bytes of object code. Each pass through the service routine requires (coincidently) 67 μs plus 32 μs once per second for the electrical test. If executed every 4 ms as suggested this software would typically reduce the throughput of the background program by less than 2%.

Once a microcomputer has been designed into a system, new features suddenly become virtually free. Software could make the emergency blinkers flash alternately or at a rate faster than the turn signals. Turn signals could override the emergency blinkers. Adding more bulbs would allow multiple tail light sequencing and syncopation—true flash factor, so to speak.

Design Example #5—Complex Control Functions

Finally, we'll mix byte and bit operations to extend the use of 8051 into extremely complex applications.

Programmers can arbitrarily assign I/O pins to input and output functions only if the total does not exceed 32, which is insufficient for applications with a very large number of input variables. One way to expand the number of inputs is with a technique similar to multiplexed-keyboard scanning.

Figure 19 shows a block diagram for a moderately complex programmable industrial controller with the following characteristics:

- 64 input variable sensors:
- 12 output signals:
- Combinational and sequential logic computations:
- Remote operation with communications to a host processor via a high-speed full-duplex serial link:
- Two prioritized external interrupts:
- Internal real-time and time-of-day clocks.

While many microprocessors could be programmed to provide these capabilities with assorted peripheral support chips, an 8051 microcomputer needs no other integrated circuits!

The 64 input sensors are logically arranged as an 8x8 matrix. The pins of Port 1 sequentially enable each column of the sensor matrix: as each is enabled Port 0 reads in the state of each sensor in that column. An eight-byte block in bit-addressable RAM remembers the data as it is read in so that after each complete scan cycle there is an internal map of the current state of all sensors. Logic functions can then directly address the elements of the bit map.

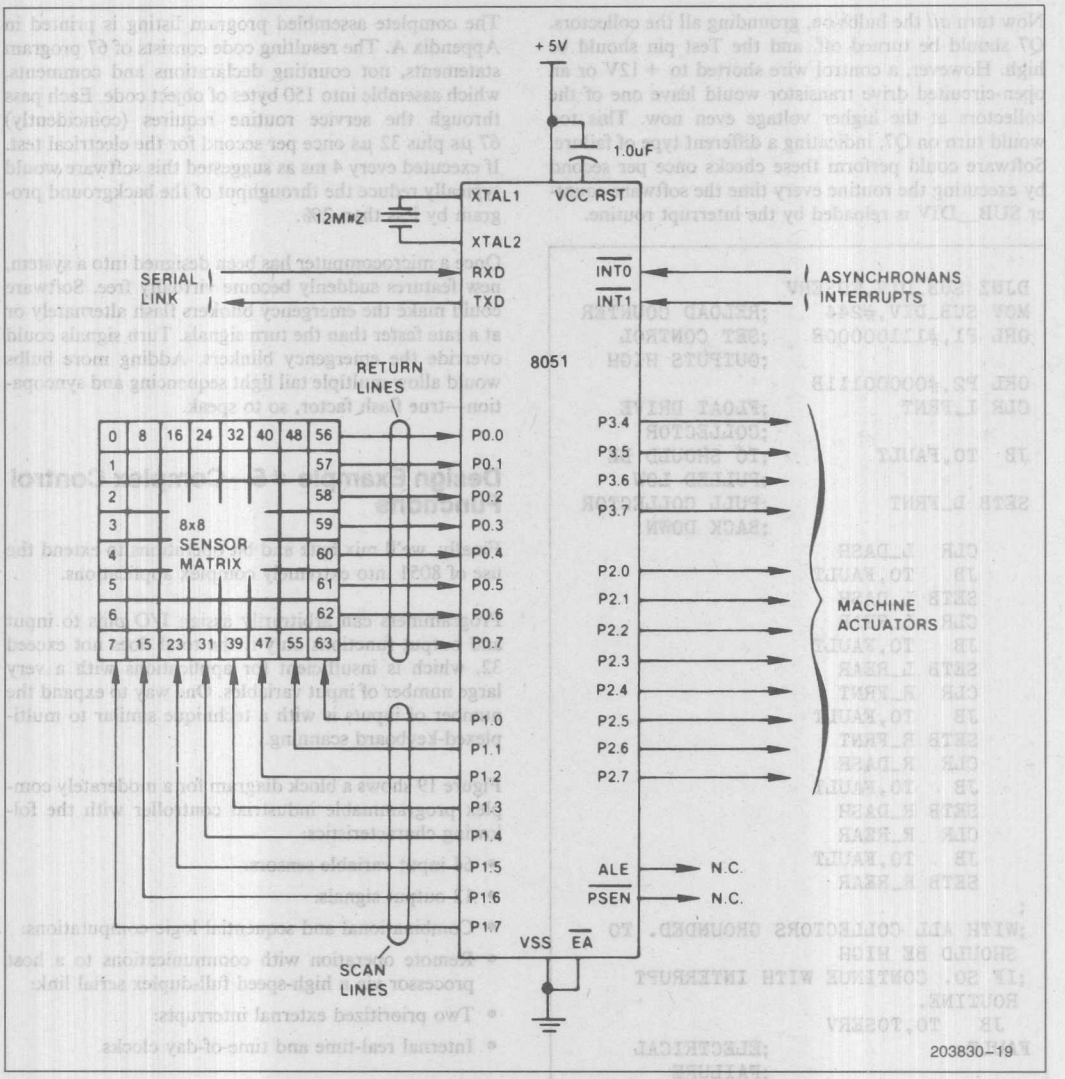


Figure 19. Block Diagram of 64-Input Machine Controller

The computer's serial port is configured as a nine-bit UART, transferring data at 17,000 bytes-per-second. The ninth bit may distinguish between address and data bytes.

The 8051 serial port can be configured to detect bytes with the address bit set, automatically ignoring all others. Pins INT0 and INT1 are interrupts configured respectively as high-priority, falling-edge triggered and low-priority, low-level triggered. The remaining 12 I/O pins output TTL-level control signals to 12 actuators.

There are several ways to implement the sensor matrix circuitry, all logically similar. Figure 20a shows one possibility. Each of the 64 sensors consists of a pair of simple switch contacts in series with a diode to permit multiple contact closures throughout the matrix.

The scan lines from Port 1 provide eight un-encoded active-high scan signals for enabling columns of the matrix. The return lines on rows where a contact is closed are pulled high and read as logic ones. Open return lines are pulled to ground by one of the 40 kΩ resistors and are read as zeroes. (The resistor values must be chosen to ensure all return lines are pulled above the 2.0V logic threshold, even in the worst-case,

where all contacts in an enabled column are closed.) Since P0 is provided open-collector outputs and high-impedance MOS inputs its input loading may be considered negligible.

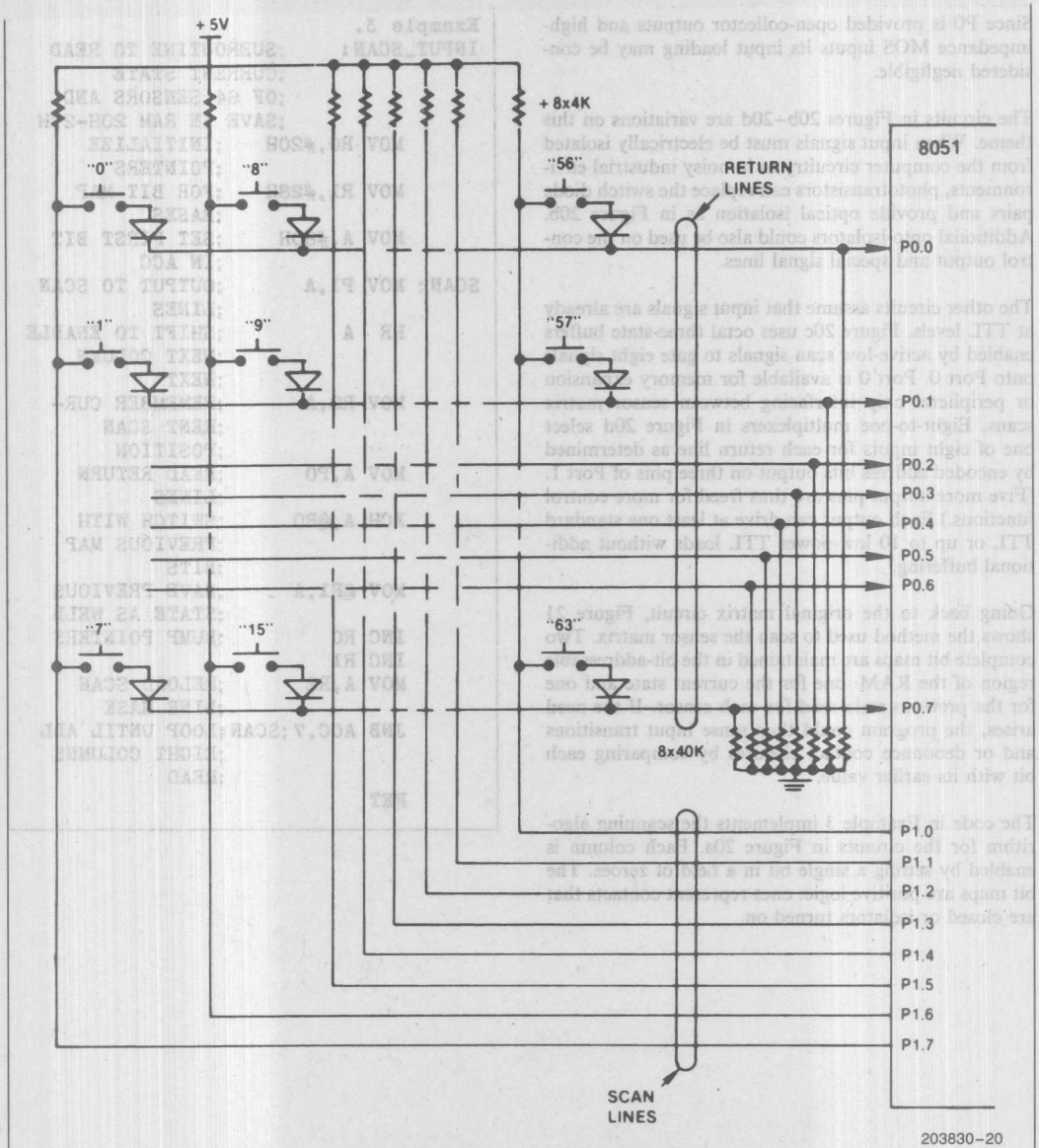
The circuits in Figures 20b–20d are variations on this theme. When input signals must be electrically isolated from the computer circuitry as in noisy industrial environments, phototransistors can replace the switch diode pairs and provide optical isolation as in Figure 20b. Additional opto-isolators could also be used on the control output and special signal lines.

The other circuits assume that input signals are already at TTL levels. Figure 20c uses octal three-state buffers enabled by active-low scan signals to gate eight signals onto Port 0. Port 0 is available for memory expansion or peripheral chip interfacing between sensor matrix scans. Eight-to-one multiplexers in Figure 20d select one of eight inputs for each return line as determined by encoded address bits output on three pins of Port 1. (Five more output pins are thus freed for more control functions.) Each output can drive at least one standard TTL or up to 10 low-power TTL loads without additional buffering.

Going back to the original matrix circuit, Figure 21 shows the method used to scan the sensor matrix. Two complete bit maps are maintained in the bit-addressable region of the RAM: one for the current state and one for the previous state read for each sensor. If the need arises, the program could then sense input transitions and or debounce contact closures by comparing each bit with its earlier value.

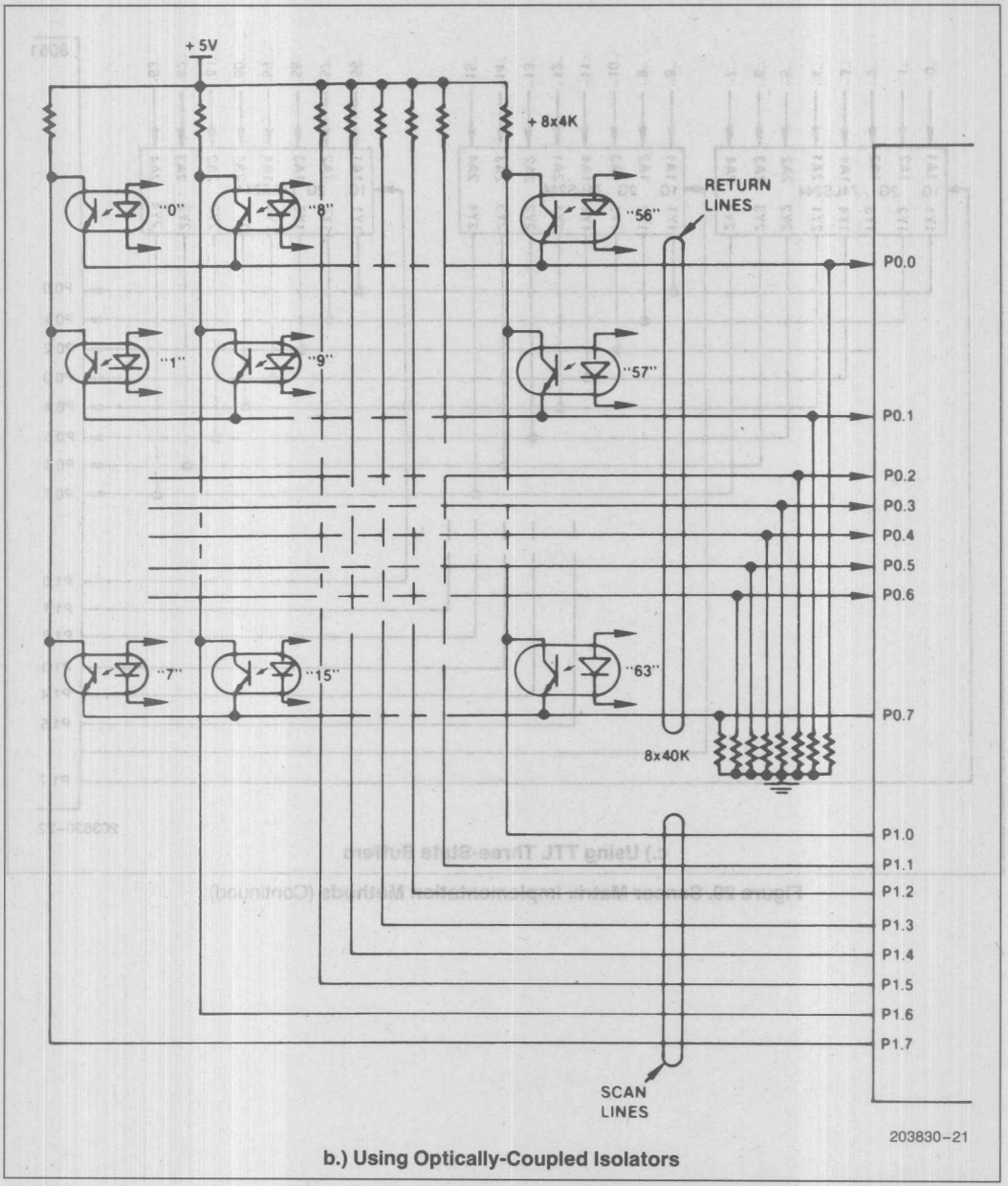
The code in Example 3 implements the scanning algorithm for the circuits in Figure 20a. Each column is enabled by setting a single bit in a field of zeroes. The bit maps are positive logic: ones represent contacts that are closed or isolators turned on.

```
Example 3.
INPUT_SCAN:     ;SUBROUTINE TO READ
                ;CURRENT STATE
                ;OF 64 SENSORS AND
                ;SAVE IN RAM 20H-27H
      MOV R0,#20H    ;INITIALIZE
                     ;POINTERS
      MOV R1,#28H    ;FOR BIT MAP
                     ;BASES
      MOV A,#80H     ;SET FIRST BIT
                     ;IN ACC
SCAN; MOV P1,A       ;OUTPUT TO SCAN
                     ;LINES
      RR  A          ;SHIFT TO ENABLE
                     ;NEXT COLUMN
                     ;NEXT
      MOV R2,A       ;REMEMBER CUR-
                     ;RENT SCAN
                     ;POSITION
      MOV A,P0       ;READ RETURN
                     ;LINES
      XCH A,@R0      ;SWITCH WITH
                     ;PREVIOUS MAP
                     ;BITS
      MOV @R1,A      ;SAVE PREVIOUS
                     ;STATE AS WELL
      INC R0         ;BUMP POINTERS
      INC R1
      MOV A,R2       ;RELOAD SCAN
                     ;LINE MASK
      JNB ACC,7;SCAN;LOOP UNTIL ALL
                     ;EIGHT COLUMNS
                     ;READ
      RET
```



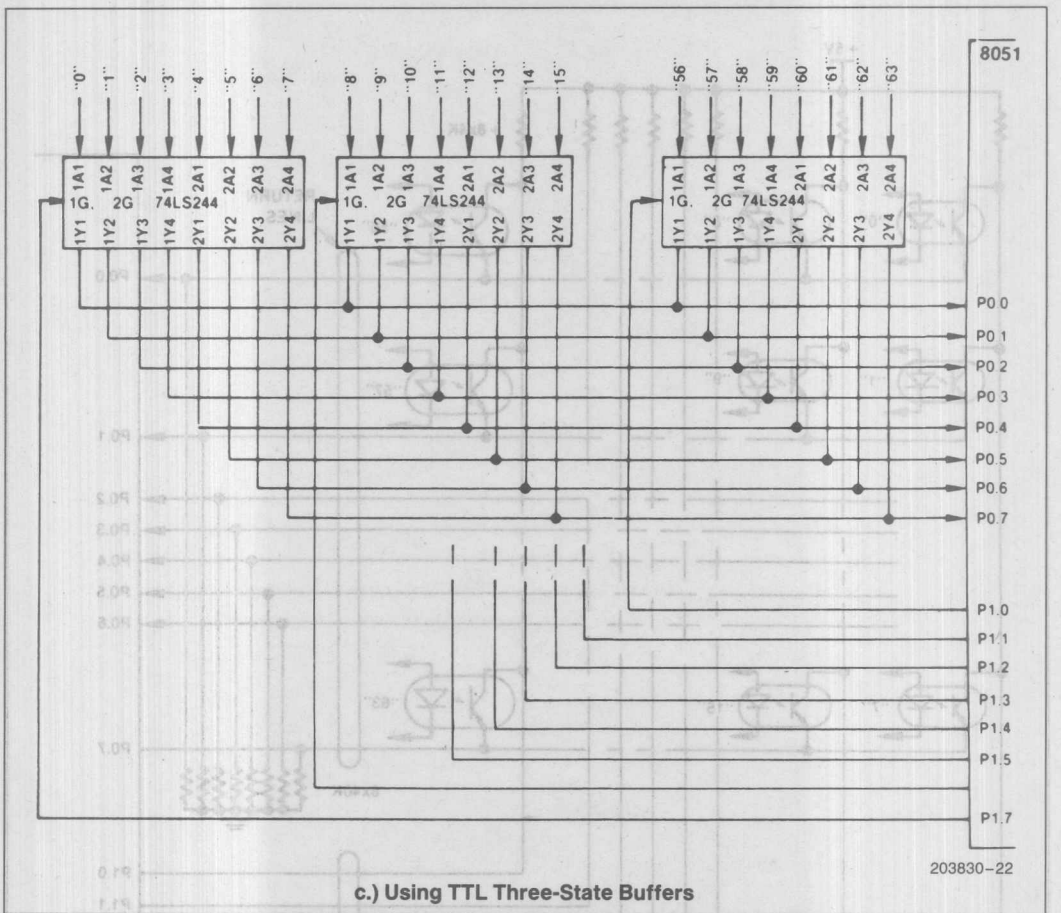
a.) Using Switch Contact/Diode Matrix

Figure 20. Sensor Matrix Implementation Methods



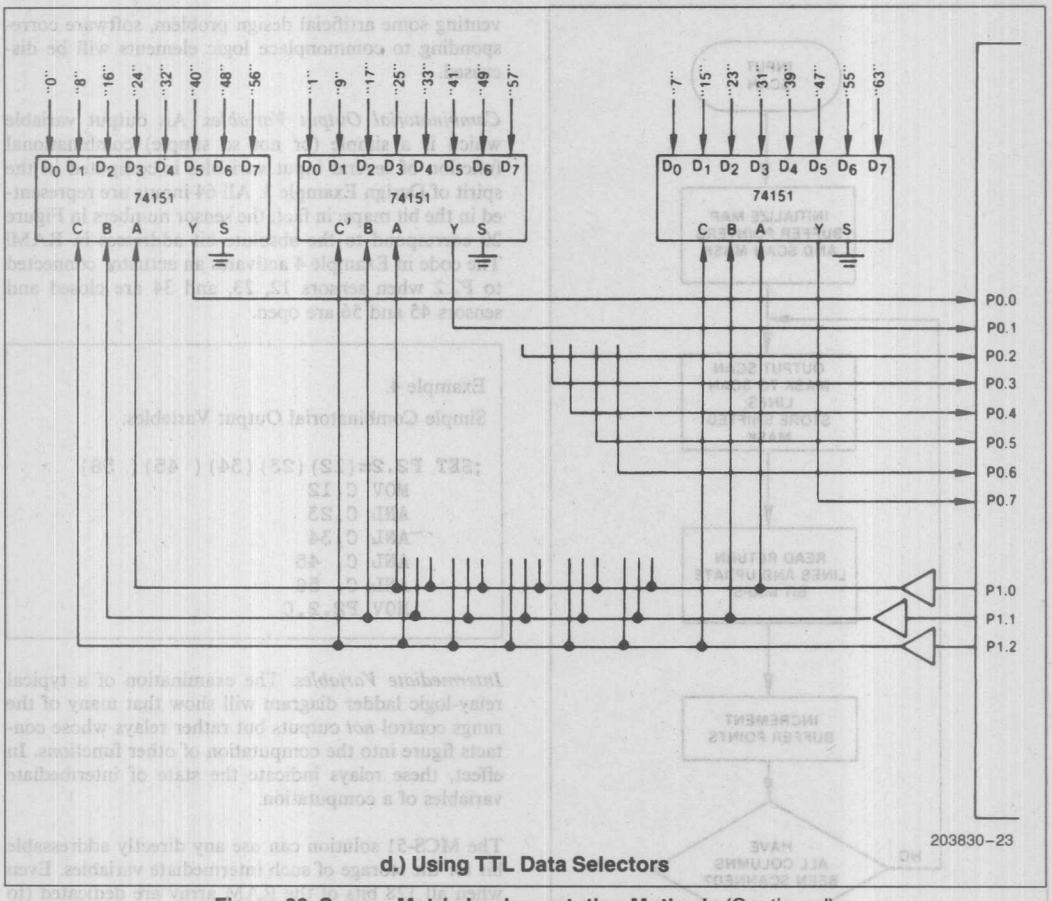
b.) Using Optically-Coupled Isolators

Figure 20. Sensor Matrix Implementation Methods (Continued)



c.) Using TTL Three-State Buffers

Figure 20. Sensor Matrix Implementation Methods (Continued)



d.) Using TTL Data Selectors

Figure 20. Sensor Matrix Implementation Methods (Continued)

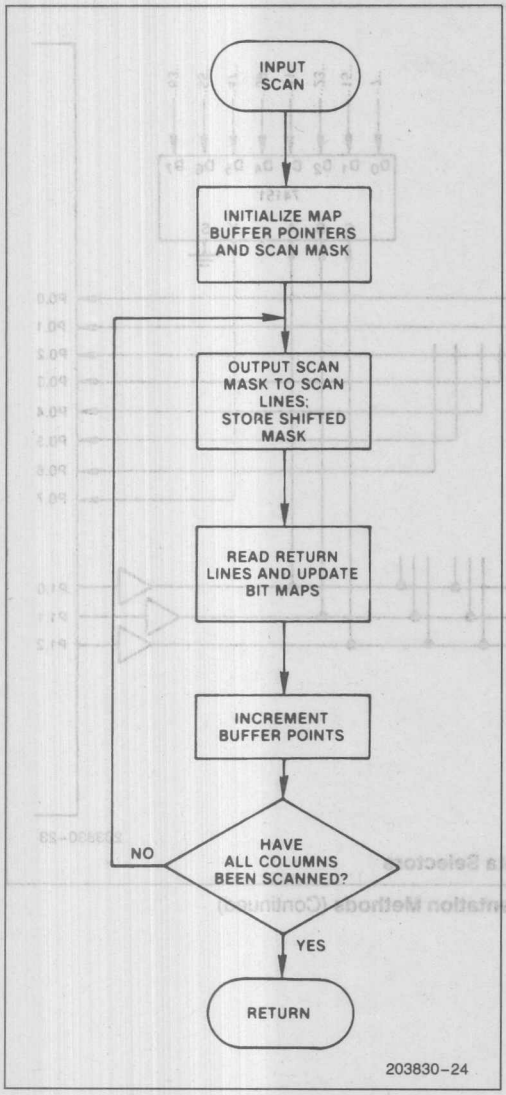


Figure 21. Flowchart for Reading in Sensor Matrix

What happens after the sensors have been scanned depends on the individual application. Rather than inventing some artificial design problem, software corresponding to commonplace logic elements will be discussed.

Combinatorial Output Variables. An output variable which is a simple (or not so simple) combinational function of several input variables is computed in the spirit of Design Example 3. All 64 inputs are represented in the bit maps: in fact, the sensor numbers in Figure 20 correspond to the absolute bit addresses in RAM! The code in Example 4 activates an actuator connected to P2.2 when sensors 12, 23, and 34 are closed and sensors 45 and 56 are open.

Example 4.
Simple Combinatorial Output Variables.

```
;SET P2.2=(12)(23)(34)( 45)( 56)
      MOV C,12
      ANL C,23
      ANL C,34
      ANL C, 45
      ANL C, 56
      MOV P2.2,C
```

Intermediate Variables. The examination of a typical relay-logic ladder diagram will show that many of the rungs control *not* outputs but rather relays whose contacts figure into the computation of other functions. In effect, these relays indicate the state of intermediate variables of a computation.

The MCS-51 solution can use any directly addressable bit for the storage of such intermediate variables. Even when all 128 bits of the RAM array are dedicated (to input bit maps in this example), the accumulator, PSW, and B register provide 18 additional flags for intermediate variables.

For example, suppose switches 0 through 3 control a safety interlock system. Closing any of them should deactivate certain outputs. Figure 22 is a ladder diagram for this situation. The interlock function could be recomputed for every output affected, or it may be computed once and save (as implied by the diagram). As the program proceeds this bit can qualify each output.

Example 5. Incorporating Override signal into actuator outputs.

```
;       CALL INPUT_SCAN
        MOV C,0
        ORL C,1
        ORL C,2
        ORL C,3
        MOV F0,C
;       ....  .....
;     COMPUTE FUNCTION 0
;
        ANL C, F0
        MOV PL0,C
;       ....  .....
;     COMPUTE FUNCTION 1
;
        ANL C, F0
        MOV P1,1,C
;       ....  .....
;     COMPUTE FUNCTION 2
;
        ANL C, F0
        MOV P1,2,C
;       ....  .....
```

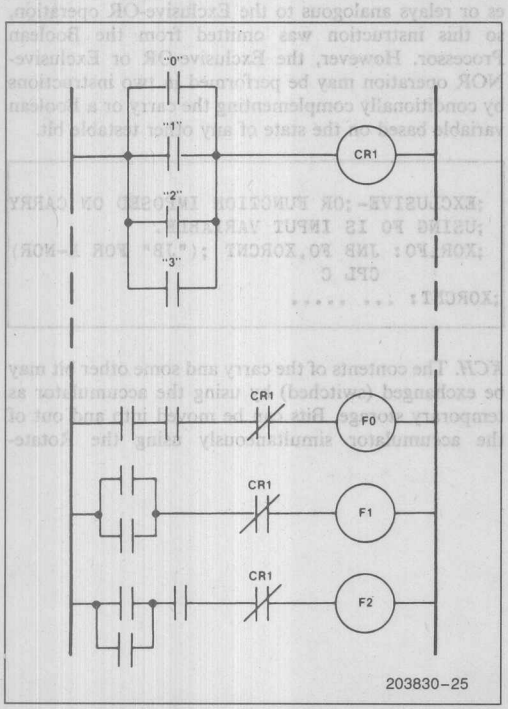


Figure 22. Ladder Diagram for Output Override Circuitry

Latching Relays. A latching relay can be forced into either the ON or OFF state by two corresponding input signals, where it will remain until forced onto the opposite state—analogous to a TTL Set/Reset flip-flop. The relay is used as an intermediate variable for other calculations. In the previous example, the emergency condition could be remembered and remain active until an "emergency cleared" button is pressed.

Any flag or addressable bit may represent a latching relay with a few lines of code (see Example 6).

Example 6. Simulating a latching relay.

```
;I_SET SET FLAG 0 IF C=1
I_SET:  ORL C,F0
        MOV F0,C
;
;I_RSET RESET FLAG 0 IF C=1
I_RSET: CPS C
        ANL C,F0
        MOV F0,C
;
```

Time Delay Relays. A time delay relay does not respond to an input signal until it has been present (or absent) for some predefined time. For example, a ballast or load resistor may be switched in series with a D.C. motor when it is first turned on, and shunted from the circuit after one second. This sort of time delay may be simulated by an interrupt routine driven by one of the two 8051 timer counters. The procedure followed by the routine depends heavily on the details of the exact function needed: time-outs or time delays with resettable or non-resettable inputs are possible. If the interrupt routine is executed every 10 milliseconds the code in Example 7 will clear an intermediate variable set by the background program after it has been active for two seconds.

Example 7. Code to clear USRFLG after a fixed time delay.

```
        JNB  USR_FLG,NXTTST
        DJNZ DLAY_COUNT,NXTTST
        CLR  USR_FLG
        MOV  DLAY_COUNT,#200
NXTTST; ;..  .....
```

emergency conditions represented by certain input combinations (such as the earlier Emergency Override), the controller could shut down the machine immediately and/or alert the host processor via the serial port. Code bytes indicating the nature of the problem could be transmitted to a central computer. In fact, at 17,000 bytes-per-second, the entire contents of both bit maps could be sent to the host processor for further analysis in less than a millisecond! If the host decides that conditions warrant, it could alert other remote processors in the system that a problem exists and specify which shut-down sequence each should initiate. For more information on using the serial port, consult the MCS-51 User's Manual.

Response Timing

One difference between relay and programmed industrial controllers (when each is considered as a "black box") is their respective reaction times to input changes. As reflected by a ladder diagram, relay systems contain a large number of "rungs" operating in parallel. A change in input conditions will begin propagating through the system immediately, possibly affecting the output state within milliseconds.

Software, on the other hand, operates sequentially. A change in input states will not be detected until the next time an input scan is performed, and will not affect the outputs until that section of the program is reached. For that reason the raw speed of computing the logical functions is of extreme importance.

Here the Boolean processor pays off. *Every instruction mentioned in this Note* completes in one or two microseconds—the *minimum* instruction execution time for many other microcontrollers! A ladder diagram containing a hundred rungs, with an average of four contacts per rung can be replaced by approximately five hundred lines of software. A complete pass through the entire matrix scanning routine and all computations would require about a millisecond: less than the time it takes for most relays to change state.

an function with a subroutine would be less efficient by at least an order of magnitude. Extra software is needed for the simulation routines, and each step takes longer to execute for three reasons: several byte-wide logical instructions are executed per user program step (rather than one Boolean operation): most of those instructions take longer to execute with microprocessors performing multiple off-chip accesses: and calling and returning from the various subroutines requires overhead for stack operations.

In fact, the speed of the Boolean Processor solution is likely to be much faster than the system requires. The CPU might use the time left over to compute feedback parameters, collect and analyze execution statistics, perform system diagnostics, and so forth.

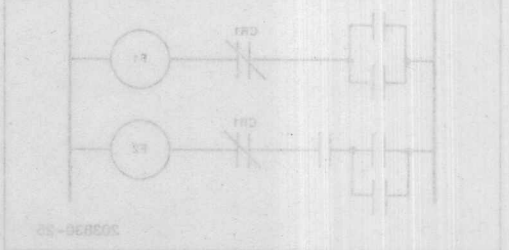
Additional Functions and Uses

With the building-block basics mentioned above many more operations may be synthesized by short instruction sequences.

Exclusive-OR. There are no common mechanical devices or relays analogous to the Exclusive-OR operation, so this instruction was omitted from the Boolean Processor. However, the Exclusive-OR or Exclusive-NOR operation may be performed in two instructions by conditionally complementing the carry or a Boolean variable based on the state of any other testable bit.

```
;EXCLUSIVE-;OR FUNCTION IMPOSED ON CARRY
;USING F0 IS INPUT VARIABLE.
;XOR_F0: JNB F0,XORCNT ;("JB" FOR X-NOR)
        CPL C
;XORCNT: ... .....
```

XCH. The contents of the carry and some other bit may be exchanged (switched) by using the accumulator as temporary storage. Bits can be moved into and out of the accumulator simultaneously using the Rotate-



through-carry instructions, though this would alter the accumulator data.

```
;EXCHANGE CARRY WITH USRFLG
XCHBIT: RLC   A
        MOV   C,USR_FLG
        RRC   A
        MOV   USR_FLG,C
        RLC   A
```

Extended Bit Addressing. The 8051 can directly address 144 general-purpose bits for all instructions in Figure 3b. Similar operations may be extended to any bit anywhere on the chip with some loss of efficiency.

The logical operations AND, OR, and Exclusive-OR are performed on byte variables using six different addressing modes, one of which lets the source be an immediate mask, and the destination any directly addressable byte. Any bit may thus be set, cleared, or complemented with a three-byte, two-cycle instruction if the mask has all bits but one set or cleared.

Byte variables, registers, and indirectly addressed RAM may be moved to a bit addressable register (usually the accumulator) in one instruction. Once transferred, the bits may be tested with a conditional jump, allowing any bit to be polled in 3 microseconds—still much faster than most architectures—or used for logical calculations. (This technique can also simulate additional bit addressing modes with byte operations.)

Parity of bytes or bits. The parity of the current accumulator contents is always available in the PSW, from whence it may be moved to the carry and further processed. Error-correcting Hamming codes and similar applications require computing parity on groups of isolated bits. This can be done by conditionally complementing the carry flag based on those bits or by gathering the bits into the accumulator (as shown in the DES example) and then testing the parallel parity flag.

Multiple byte shift and CRC codes

Though the 8051 serial port can accommodate eight- or nine-bit data transmissions, some protocols involve much longer bit streams. The algorithms presented in Design Example 2 can be extended quite readily to 16 or more bits by using multi-byte input and output buffers.

Many mass data storage peripherals and serial communications protocols include Cyclic Redundancy (CRC) codes to verify data integrity. The function is generally computed serially by hardware using shift registers and Exclusive-OR gates, but it can be done with software. As each bit is received into the carry, appropriate bits in the multi-byte data buffer are conditionally complemented based on the incoming data bit. When finished, the CRC register contents may be checked for zero by ORing the two bytes in the accumulator.

4.0 SUMMARY

A truly unique facet of the Intel MCS-51 microcomputer family design is the collection of features optimized for the one-bit operations so often desired in real-world, real-time control applications. Included are 17 special instructions, a Boolean accumulator, implicit and direct addressing modes, program and mass data storage, and many I/O options. These are the world's first single-chip microcomputers able to efficiently manipulate, operate on, and transfer either bytes or individual bits as data.

This Application Note has detailed the information needed by a microcomputer system designer to make full use of these capabilities. Five design examples were used to contrast the solutions allowed by the 8051 and those required by previous architectures. Depending on the individual application, the 8051 solution will be easier to design, more reliable to implement, debug, and verify, use less program memory, and run up to an order of magnitude faster than the same function implemented on previous digital computer architectures.

Combining byte- and bit-handling capabilities in a single microcomputer has a strong synergistic effect: the power of the result exceeds the power of byte- and bit-processors laboring individually. Virtually all user applications will benefit in some way from this duality. Data intensive applications will use bit addressing for test pin monitoring or program control flags: control applications will use byte manipulation for parallel I/O expansion or arithmetic calculations.

It is hoped that these design examples give the reader an appreciation of these unique features and suggest ways to exploit them in his or her own application.

APPENDIX A
Automobile Turn-Indicator Controller Program Listing

```
ISIS-II MCS-51 MACRO ASSEMBLER V1.0
OBJECT MODULE PLACED IN :F0:AP70.HEX
ASSEMBLER INVOKED BY:  :f1:asm51 ap70 src date(328)
LOC  OBJ            LINE     SOURCE

                       1     $XREF TITLE(AP-70 APPENDIX)
                       2     ;*******************************************************
                       3     ;
                       4     ;       THE FOLLOWING PROGRAM USES THE BOOLEAN INSTRUCTION SET
                       5     ;       OF THE INTEL 8051 MICROCOMPUTER TO PERFORM A NUMBER OF
                       6     ;       AUTOMOTIVE DASHBOARD CONTROL FUNCTIONS RELATING TO
                       7     ;       TURN SIGNAL CONTROL, EMERGENCY BLINKERS, BRAKE LIGHT
                       8     ;       CONTROL, AND PARKING LIGHT OPERATION.
                       9     ;       THE ALGORITHMS AND HARDWARE ARE DESCRIBED IN DESIGN
                      10     ;       EXAMPLE #4 OF INTEL APPLICATION NOTE AP-70,
                      11     ;                "USING THE INTEL MCS-51(TM)
                      12     ;              BOOLEAN PROCESSING CAPABILITIES"
                      13     ;
                      14     ;*******************************************************
                      15     ;
                      16     ;       INPUT PIN DECLARATIONS:
                      17     ;        (ALL INPUTS ARE POSITIVE-TRUE LOGIC
                      18     ;       INPUTS ARE HIGH WHEN RESPECTIVE SWITCH CONTACT IS CLOSED )
                      19     ;
0090                  20     BRAKE   BIT     P1.0    ; BRAKE PEDAL DEPRESSED
0091                  21     EMERG   BIT     P1.1    ; EMERGENCY BLINKER ACTIVATED
0092                  22     PARK    BIT     P1.2    ; PARKING LIGHTS ON
0093                  23     L_TURN  BIT     P1.3    ; TURN LEVER DOWN
0094                  24     R_TURN  BIT     P1.4    ; TURN LEVER UP
                      25     ;
                      26     ;       OUTPUT PIN DECLARATIONS:
                      27     ;       (ALL OUTPUTS ARE POSITIVE TRUE LOGIC
                      28     ;       BULB IS TURNED ON WHEN OUTPUT PIN IS HIGH.)
                      29     ;
0095                  30     L_FRNT  BIT     P1.5    ; FRONT LEFT-TURN INDICATOR
0096                  31     R_FRNT  BIT     P1.6    ; FRONT RIGHT-TURN INDICATOR
0097                  32     L_DASH  BIT     P1.7    ; DASHBOARD LEFT-TURN INDICATOR
00A0                  33     R_DASH  BIT     P2.0    ; DASHBOARD RIGHT-TURN INDICATOR
00A1                  34     L_REAR  BIT     P2.1    ; REAR LEFT-TURN INDICATOR
00A2                  35     R_REAR  BIT     P2.2    ; REAR RIGHT-TURN INDICATOR
                      36     ;
00A3                  37     S_FAIL  BIT     P2.3    ; ELECTRICAL SYSTEM FAULT INDICATOR
                      38     ;
                      39     ;       INTERNAL VARIABLE DEFINITIONS:
                      40     ;
0020                  41     SUB_DIV DATA    20H             ; INTERRUPT RATE SUBDIVIDER
0000                  42     HI_FREQ BIT     SUB_DIV.0       ; HIGH-FREQUENCY OSCILLATOR BIT
0007                  43     LO_FREQ BIT     SUB_DIV.7       ; LOW-FREQUENCY OSCILLATOR BIT
                      44     ;
00D1                  45     DIM     BIT     PSW.1           ; PARKING LIGHTS ON FLAG
                      46     ;
                      47     ;=======================================================
                     48 +1   $EJECT
```

203830-26

```
LOC  OBJ             LINE     SOURCE

                       49             ORG     0000H           ; RESET VECTOR
0000 020040            50             LJMP    INIT
                       51     ;
000B                   52             ORG     000BH           ; TIMER 0 SERVICE VECTOR
000B 758CF0            53             MOV     TH0,#-16        ; HIGH TIMER BYTE ADJUSTED TO CONTROL INT  RATE
000E C0D0              54             PUSH    PSW             ; EXECUTE CODE TO SAVE ANY REGISTERS USED BELOW
0010 0154              55             AJMP    UPDATE          ;    (CONTINUE WITH REST OF ROUTINE)
                       56     ;
0040                   57             ORG     0040H
0040 758A00            58     INIT    MOV     TL0,#0          ; ZERO LOADED INTO LOW-ORDER BYTE AND
0043 758CF0            59             MOV     TH0,#-16        ; -16 IN HIGH-ORDER BYTE GIVES 4 MSEC PERIOD
0046 758961            60             MOV     TMOD,#01100001B ; 8-BIT AUTO RELOAD COUNTER MODE FOR TIMER 1,
                       61                                     ;  16-BIT TIMER MODE FOR TIMER 0 SELECTED
0049 7520F4            62             MOV     SUB_DIV,#244    ; SUBDIVIDE INTERRUPT RATE BY 244 FOR 1 HZ
004C D2A9              63             SETB    ET0             ; USE TIMER 0 OVERFLOWS TO INTERRUPT PROGRAM
004E D2AF              64             SETB    EA              ; CONFIGURE IE TO GLOBALLY ENABLE INTERRUPTS
0050 D28C              65             SETB    TR0             ; KEEP INSTRUCTION CYCLE COUNT UNTIL OVERFLOW
0052 80FE              66             SJMP    $               ; START BACKGROUND PROGRAM EXECUTION
                       67     ;
                       68     ;
0054 D52038            69     UPDATE  DJNZ    SUB_DIV,TOSERV  ; EXECUTE SYSTEM TEST ONLY ONCE PER SECOND
0057 7520F4            70             MOV     SUB_DIV,#244    ; GET VALUE FOR NEXT ONE SECOND DELAY AND
                       71     ;                               ; GO THROUGH ELECTRICAL SYSTEM TEST CODE
005A 4390E0            72             ORL     P1,#11100000B   ; SET CONTROL OUTPUTS HIGH
005D 43A007            73             ORL     P2,#00000111B
0060 C295              74             CLR     L_FRNT          ; FLOAT DRIVE COLLECTOR
0062 20B428            75             JB      T0,FAULT        ; T0 SHOULD BE PULLED LOW
0065 D295              76             SETB    L_FRNT          ; PULL COLLECTOR BACK DOWN
0067 C297              77             CLR     L_DASH          ; REPEAT SEQUENCE FOR L_DASH,
0069 20B421            78             JB      T0,FAULT
006C D297              79             SETB    L_DASH
006E C2A1              80             CLR     L_REAR          ;               L_REAR,
0070 20B41A            81             JB      T0,FAULT
0073 D2A1              82             SETB    L_REAR
0075 C296              83             CLR     R_FRNT          ;            R_FRNT,
0077 20B413            84             JB      T0,FAULT
007A D296              85             SETB    R_FRNT
007C C2A0              86             CLR     R_DASH          ;            R_DASH,
007E 20B40C            87             JB      T0,FAULT
0081 D2A0              88             SETB    R_DASH
0083 C2A2              89             CLR     R_REAR          ;              AND R_REAR
0085 20B405            90             JB      T0,FAULT
0088 D2A2              91             SETB    R_REAR
                       92     ;
                       93     ;       WITH ALL COLLECTORS GROUNDED, T0 SHOULD BE HIGH
                       94     ;       IF SO, CONTINUE WITH INTERRUPT ROUTINE
                       95     ;
008A 20B402            96             JB      T0,TOSERV
008D B2A3              97     FAULT   CPL     S_FAIL          ; ELECTRICAL FAILURE PROCESSING ROUTINE
                       98                                     ; (TOGGLE INDICATOR ONCE PER SECOND)
                       99 +1  $EJECT
```

203830-27

```
LOC  OBJ            LINE     SOURCE

                     100      ;       CONTINUE WITH INTERRUPT PROCESSING.
                     101      ;
                     102      ;   1)  COMPUTE LOW BULB INTENSITY WHEN PARKING LIGHTS ARE ON.
                     103      ;
008F A201            104      TOSERV: MOV     C,SUB_DIV 1     ; START WITH 50 PERCENT,
0091 8200            105              ANL     C,SUB_DIV 0     ; MASK DOWN TO 25 PERCENT,
0093 7202            106              ORL     C,SUB_DIV 2     ; BUILD BACK TO 62.5 PERCENT,
0095 8292            107              ANL     C,PARK          ; GATE WITH PARKING LIGHT SWITCH,
0097 92D1            108              MOV     DIM,C           ; AND SAVE IN TEMP. VARIABLE.
                     109      ;
                     110      ;   2)  COMPUTE AND OUTPUT LEFT-HAND DASHBOARD INDICATOR.
                     111      ;
0099 A293            112              MOV     C,L_TURN        ; SET CARRY IF TURN
009B 7291            113              ORL     C,EMERG         ; OR EMERGENCY SELECTED.
009D 8207            114              ANL     C,LO_FREQ       ; IF SO, GATE IN 1 HZ SIGNAL
009F 9297            115              MOV     L_DASH,C        ; AND OUTPUT TO DASHBOARD.
                     116      ;
                     117      ;   3)  COMPUTE AND OUTPUT LEFT-HAND FRONT TURN SIGNAL.
                     118      ;
00A1 92D5            119              MOV     F0,C            ; SAVE FUNCTION SO FAR.
00A3 72D1            120              ORL     C,DIM           ; ADD IN PARKING LIGHT FUNCTION
00A5 9295            121              MOV     L_FRNT,C        ; AND OUTPUT TO TURN SIGNAL.
                     122      ;
                     123      ;   4)  COMPUTE AND OUTPUT LEFT-HAND REAR TURN SIGNAL.
                     124      ;
00A7 A290            125              MOV     C,BRAKE         ; GATE BRAKE PEDAL SWITCH
00A9 B093            126              ANL     C,/L_TURN       ; WITH TURN LEVER.
00AB 72D5            127              ORL     C,F0            ; INCLUDE TEMP. VARIABLE FROM DASH
00AD 72D1            128              ORL     C,DIM           ; AND PARKING LIGHT FUNCTION
00AF 92A1            129              MOV     L_REAR,C        ; AND OUTPUT TO TURN SIGNAL.
                     130      ;
                     131      ;   5)  REPEAT ALL OF ABOVE FOR RIGHT-HAND COUNTERPARTS.
                     132      ;
00B1 A294            133              MOV     C,R_TURN        ; SET CARRY IF TURN
00B3 7291            134              ORL     C,EMERG         ; OR EMERGENCY SELECTED.
00B5 8207            135              ANL     C,LO_FREQ       ; IF SO, GATE IN 1 HZ SIGNAL
00B7 92A0            136              MOV     R_DASH,C        ; AND OUTPUT TO DASHBOARD.
00B9 92D5            137              MOV     F0,C            ; SAVE FUNCTION SO FAR.
00BB 72D1            138              ORL     C,DIM           ; ADD IN PARKING LIGHT FUNCTION
00BD 9296            139              MOV     R_FRNT,C        ; AND OUTPUT TO TURN SIGNAL.
00BF A290            140              MOV     C,BRAKE         ; GATE BRAKE PEDAL SWITCH
00C1 B094            141              ANL     C,/R_TURN       ; WITH TURN LEVER
00C3 72D5            142              ORL     C,F0            ; INCLUDE TEMP. VARIABLE FROM DASH
00C5 72D1            143              ORL     C,DIM           ; AND PARKING LIGHT FUNCTION
00C7 92A2            144              MOV     R_REAR,C        ; AND OUTPUT TO TURN SIGNAL.
                     145      ;
                     146      ;       RESTORE STATUS REGISTER AND RETURN.
                     147      ;
00C9 D0D0            148              POP     PSW             ; RESTORE PSW
00CB 32              149              RETI                    ; AND RETURN FROM INTERRUPT ROUTINE
                     150      ;
                     151              END
```

203830-28

```
XREF SYMBOL TABLE LISTING
---- ------ ----- -------


NAME     TYPE     VALUE AND REFERENCES

BRAKE    N BSEG   0090H  20# 125 140
DIM      N BSEG   00D1H  45# 108 120 128 138 143
EA       N BSEG   00AFH  64
EMERG    N BSEG   0091H  21# 113 134
ETO      N BSEG   00A9H  63
FO       N BSEG   00D5H  119 127 137 142
FAULT    L CSEG   008DH  75 78 81 84 87 90 97#
HI_FREQ  N BSEG   0000H  42#
INIT     L CSEG   0040H  50 58#
L_DASH   N BSEG   0097H  32# 77 79 115
L_FRNT   N BSEG   0095H  30# 74 76 121
L_REAR   N BSEG   00A1H  34# 80 82 129
L_TURN   N BSEG   0093H  23# 112 126
LO_FREQ  N BSEG   0007H  43# 114 135
P1       N DSEG   0090H  20 21 22 23 24 30 31 32 72
P2       N DSEG   00A0H  33 34 35 37 73
PARK     N BSEG   0092H  22# 107
PSW      N DSEG   00D0H  45 54 148
R_DASH   N BSEG   00A0H  33# 86 88 136
R_FRNT   N BSEG   0096H  31# 83 85 139
R_REAR   N BSEG   00A2H  35# 89 91 144
R_TURN   N BSEG   0094H  24# 133 141
S_FAIL   N BSEG   00A3H  37# 97
SUB_DIV  N DSEG   0020H  41# 42 43 62 69 70 104 105 106
TO       N BSEG   00B4H  75 78 81 84 87 90 96
TOSERV   L CSEG   008FH  69 96 104#
THO      N DSEG   008CH  53 59
TLO      N DSEG   008AH  58
TMOD     N DSEG   0089H  60
TRO      N BSEG   008CH  65
UPDATE   L CSEG   0054H  55 69#


ASSEMBLY COMPLETE, NO ERRORS FOUND
```

203830-29

intel

APPLICATION NOTE

AP-125

November 1986

Designing Microcontroller Systems for Electrically Noisy Environments

TOM WILLIAMSON
MCO APPLICATIONS ENGINEER

Order Number: 210313-002

Digital circuits are often thought of as being immune to noise problems, but really they're not. Noises in digital systems produce software upsets: program jumps to apparently random locations in memory. Noise-induced glitches in the signal lines can cause such problems, but the supply voltage is more sensitive to glitches than the signal lines.

Severe noise conditions, those involving electrostatic discharges, or as found in automotive environments, can do permanent damage to the hardware. Electrostatic discharges can blow a crater in the silicon. In the automotive environment, in ordinary operation, the "12V" power line can shown + and −400V transients.

This Application Note describes some electrical noises and noise environments. Design considerations, along the lines of PCB layout, power supply distribution and decoupling, and shielding and grounding techniques, that may help minimize noise susceptibility are reviewed. Special attention is given to the automotive and ESD environments.

Symptoms of Noise Problems

Noise problems are not usually encountered during the development phase of a microcontroller system. This is because benches rarely simulate the system's intended environment. Noise problems tend not to show up until the system is installed and operating in its intended environment. Then, after a few minutes or hours of normal operation the system finds itself someplace out in left field. Inputs are ignored and outputs are gibberish. The system may respond to a reset, or it may have to be turned off physically and then back on again, at which point it commences operating as though nothing had happened. There may be an obvious cause, such as an electrostatic discharge from somebody's finger to a keyboard or the upset occurs every time a copier machine is turned on or off. Or there may be no obvious cause, and nothing the operator can do will make the upset repeat itself. But a few minutes, or a few hours, or a few days later it happens again.

One symptom of electrical noise problems is randomness, both in the occurrence of the problem and in what the system does in its failure. All operational upsets that occur at seemingly random intervals are not necessarily caused by noise in the system. Marginal VCC, inadequate decoupling, rarely encountered software conditions, or timing coincidences can produce upsets that seem to occur randomly. On the other hand, some noise sources can produce upsets downright periodically. Nevertheless, the more difficult it is to characterize an upset as to cause and effect, the more likely it is to be a noise problem.

Types and Sources of Electrical Noise

The name given to electrical noises other than those that are inherent in the circuit components (such as thermal noise) is EMI: electromagnetic interference. Motors, power switches, fluorescent lights, electrostatic discharges, etc., are sources of EMI. There is a veritable alphabet soup of EMI types, and these are briefly described below.

SUPPLY LINE TRANSIENTS

Anything that switches heavy current loads onto or off of AC or DC power lines will cause large transients in these power lines. Switching an electric typewriter on or off, for example, can put a 1000V spike onto the AC power lines.

The basic mechanism behind supply line transients is shown in Figure 1. The battery represents any power source, AC or DC. The coils represent the line inductance between the power source and the switchable loads R1 and R2. If both loads are drawing current, the line current flowing through the line inductance establishes a magnetic field of some value. Then, when one of the loads is switched off, the field due to that component of the line current collapses, generating transient voltages, $v = L(di/dt)$, which try to maintain the current at its original level. That's called an "inductive kick." Because of contact bounce, transients are generated whether the switch is being opened or closed, but they're worse when the switch is being opened.

An inductive kick of one type or another is involved in most line transients, including those found in the automotive environment. Other mechanisms for line transients exist, involving noise pickup on the lines. The noise voltages are then conducted to a susceptible circuit right along with the power.

EMP AND RFI

Anything that produces arcs or sparks will radiate electromagnetic pulses (EMP) or radio-frequency interference (RFI).

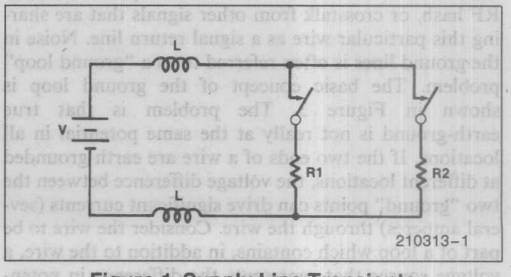


Figure 1. Supply Line Transients

Spark discharges have probably caused more software upsets in digital equipment than any other single noise source. The upsetting mechanism is the EMP produced by the spark. The EMP induces transients in the circuit, which are what actually cause the upset.

Arcs and sparks occur in automotive ignition systems, electric motors, switches, static discharges, etc. Electric motors that have commutator bars produce an arc as the brushes pass from one bar to the next. DC motors and the "universal" (AC/DC) motors that are used to power hand tools are the kinds that have commutator bars. In switches, the same inductive kick that puts transients on the supply lines will cause an opening or closing switch to throw a spark.

ESD

Electrostatic discharge (ESD) is the spark that occurs when a person picks up a static charge from walking across a carpet, and then discharges it into a keyboard, or whatever else can be touched. Walking across a carpet in a dry climate, a person can accumulate a static voltage of 35kV. The current pulse from an electrostatic discharge has an extremely fast risetime — typically, 4A/ns. Figure 2 shows ESD waveforms that have been observed by some investigators of ESD phenomena.

It is enlightening to calculate the L(di/dt) voltage required to drive an ESD current pulse through a couple of inches of straight wire. Two inches of straight wire has about 50 nH of inductance. That's not very much, but using 50 nH for L and 4A/ns for di/dt gives an L(di/dt) drop of about 200V. Recent observations by W.M. King suggest even faster risetimes (Figure 2b) and the occurrence of multiple discharges during a single discharge event.

Obviously, ESD-sensitivity needs to be considered in the design of equipment that is going to be subjected to it, such as office equipment.

GROUND NOISE

Currents in ground lines are another source of noise. These can be 60 Hz currents from the power lines, or RF hash, or crosstalk from other signals that are sharing this particular wire as a signal return line. Noise in the ground lines is often referred to as a "ground loop" problem. The basic concept of the ground loop is shown in Figure 3. The problem is that true earth-ground is not really at the same potential in all locations. If the two ends of a wire are earth-grounded at different locations, the voltage difference between the two "ground" points can drive significant currents (several amperes) through the wire. Consider the wire to be part of a loop which contains, in addition to the wire, a voltage source that represents the difference in potential between the two ground points, and you have the classical "ground loop." By extension, the term is used to refer to any unwanted (and often unexpected) currents in a ground line.

"Radiated" and "Conducted" Noise

Radiated noise is noise that arrives at the victim circuit in the form of electromagnetic radiation, such as EMP and RFI. It causes trouble by inducing extraneous voltages in the circuit. Conducted noise is noise that arrives at the victim circuit already in the form of an extraneous voltage, typically via the AC or DC power lines.

One defends against radiated noise by care in designing layouts and the use of effective shielding techniques. One defends against conducted noise with filters and

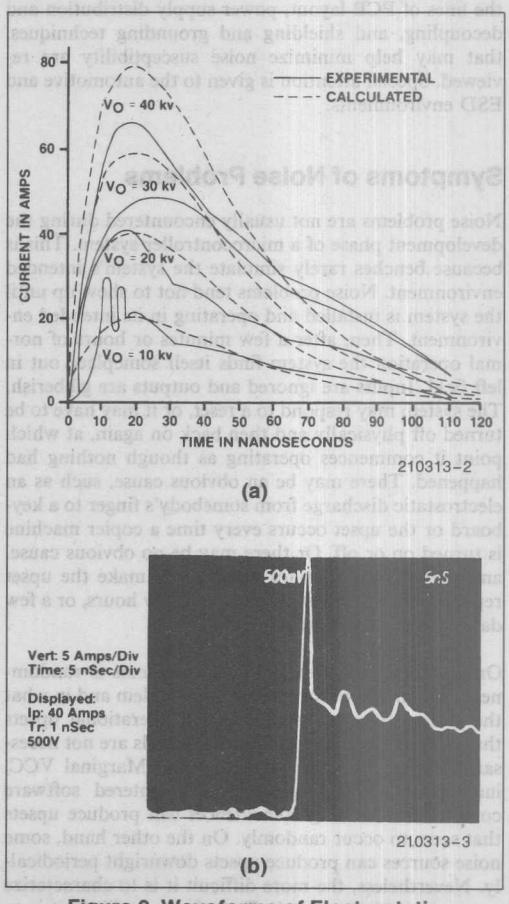


Figure 2. Waveforms of Electrostatic Discharge Currents From a Hand-Held Metallic Object

suppressors, although layouts and grounding techniques are important here, too.

Simulating the Environment

Addressing noise problems after the design of a system has been completed is an expensive proposition. The ill will generated by failures in the field is not cheap either. It's cheaper in the long run to invest a little time and money in learning about noise and noise simulation equipment, so that controlled tests can be made on the bench as the design is developing.

Simulating the intended noise environment is a two-step process: First you have to recognize what the noise environment is, that is, you have to know what kinds of electrical noises are present, and which of them are going to cause trouble. Don't ignore this first step, because it's important. If you invest in an induction coil spark generator just because your application is automotive, you'll be straining at the gnat and swallowing the camel. Spark plug noise is the least of your worries in that environment.

The second step is to generate the electrical noise in a controlled manner. This is usually more difficult than first imagined; one first imagines the simulation in terms of a waveform generator and a few spare parts, and then finds that a wideband power amplifier with a 200V dynamic range is also required. A good source of information on who supplies what noise-simulating equipment is the 1981 "ITEM" Directory and Design Guide (Reference 6).

Types of Failures and Failure Mechanisms

A major problem that EMI can cause in digital systems is intermittent operational malfunction. These software upsets occur when the system is in operation at the time an EMI source is activated, and are usually characterized by a loss of information or a jump in the execution of the program to some random location in memory. The person who has to iron out such problems is tempted to say the program counter went crazy. There is usually no damage to the hardware, and normal operation can resume as soon as the EMI has passed or the source is de-activated. Resuming normal operation usually requires manual or automatic reset, and possibly re-entering of lost information.

Electrostatic discharges from operating personnel can cause not only software upsets, but also permanent ("hard") damage to the system. For this to happen the system doesn't even have to be in operation. Sometimes the permanent damage is latent, meaning the initial damage may be marginal and require further aggravation through operating stress and time before permanent failure takes place. Sometimes too the damage is hidden.

One ESD-related failure mechanism that has been identified has to do with the bias voltage on the substrate of the chip. On some CPU chips the substrate is held at −2.5V by a phase-shift oscillator working into a capacitor/diode clamping circuit. This is called a "charge pump" in chip-design circles. If the substrate wanders too far in either direction, program read errors are noted. Some designs have been known to allow electrostatic discharge currents to flow directly into port pins of an 8048. The resulting damage to the oxide causes an increase in leakage current, which loads down the charge pump, reducing the substrate voltage to a marginal or unacceptable level. The system is then unreliable or completely inoperative until the CPU chip is replaced. But if the CPU chip was subjected to a discharge spark once, it will eventually happen again.

Chips that have a grounded substrate, such as the 8748, can sometimes sustain some oxide damage without actually becoming inoperative. In this case the damage is present, and the increased leakage current is noted; however, since the substrate voltage retains its design value, the damage is largely hidden.

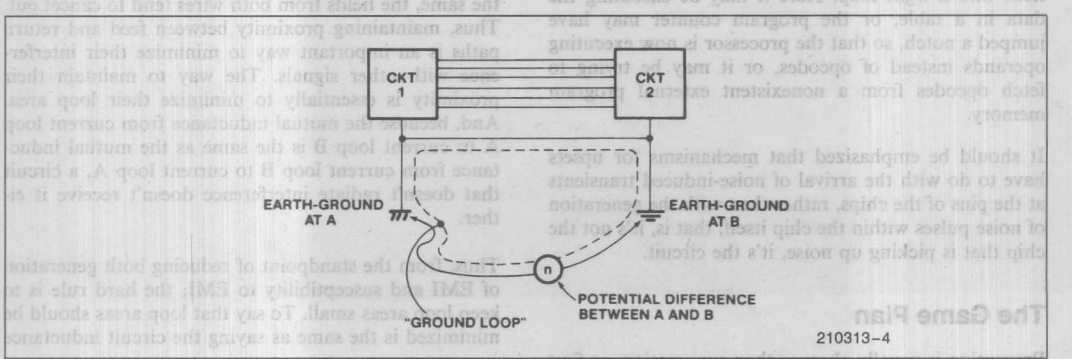


Figure 3. What a Ground Loop Is

It must therefore be recognized that connecting port pins unprotected to a keyboard or to anything else that is subject to electrostatic discharges, makes an extremely dangerous configuration. It doesn't make any difference what CPU chip is being used, or who makes it. If it connects unprotected to a keyboard, it will eventually be destroyed. Designing for an ESD-environment will be discussed further on.

We might note here that MOS chips are not the only components that are susceptible to permanent ESD damage. Bipolar and linear chips can also be damaged in this way. PN junctions are subject to a hard failure mechanism called thermal secondary breakdown, in which a current spike, such as from an electrostatic discharge, causes microscopically localized spots in the junction to approach melt temperatures. Low power TTL chips are subject to this type of damage, as are op-amps. Op-amps, in addition, often carry on-chip MOS capacitors which are directly across an external pin combination, and these are susceptible to dielectric breakdown.

We return now to the subject of software upsets. Noise transients can upset the chip through any pin, even an output pin, because every pin on the chip connects to the substrate through a pn junction. However, the most vulnerable pin is probably the VCC line, since it has direct access to all parts of the chip: every register, gate, flip-flop and buffer.

The menu of possible upset mechanisms is quite lengthy. A transient on the substrate at the wrong time will generally cause a program read error. A false level at a control input can cause an extraneous or misdirected opcode fetch. A disturbance on the supply line can flip a bit in the program counter or instruction register. A short interruption or reversal of polarity on the supply line can actually turn the processor off, but not long enough for the power-up reset capacitor to discharge. Thus when the transient ends, the chip starts up again without a reset.

A common failure mode is for the processor to lock itself into a tight loop. Here it may be executing the data in a table, or the program counter may have jumped a notch, so that the processor is now executing operands instead of opcodes, or it may be trying to fetch opcodes from a nonexistent external program memory.

It should be emphasized that mechanisms for upsets have to do with the arrival of noise-induced transients at the pins of the chips, rather than with the generation of noise pulses within the chip itself, that is, it's not the chip that is picking up noise, it's the circuit.

The Game Plan

Prevention is usually cheaper than suppression, so first we'll consider some preventive methods that might help to minimize the generation of noise voltages in the circuit. These methods involve grounding, shielding, and wiring techniques that are directed toward the mechanisms by which noise voltages are generated in the circuit. We'll also discuss methods of decoupling. Then we'll look at some schemes for making a graceful recovery from upsets that occur in spite of preventive measures. Lastly, we'll take another look at two special problem areas: electrostatic discharges and the automotive environment.

Current Loops

The first thing most people learn about electricity is that current won't flow unless it can flow in a closed loop. This simple fact is sometimes temporarily forgotten by the overworked engineer who has spent the past several years mastering the intricacies of the DO loop, the timing loop, the feedback loop, and maybe even the ground loop. The simple current loop probably owes its apparent demise to the invention of the ground symbol. By a stroke of the pen one avoids having to draw the return paths of most of the current loops in the circuit. Then "ground" turns into an infinite current sink, so that any current that flows into it is gone and forgotten. Forgotten it may be, but it's not gone. It must return to its source, so that its path will by all the laws of nature form a closed loop.

The physical geometry of a given current loop is the key to why it generates EMI, why it's susceptible to EMI, and how to shield it. Specifically, it's the area of the loop that matters.

Any flow of current generates a magnetic field whose intensity varies inversely to the distance from the wire that carries the current. Two parallel wires conducting currents $+I$ and $-I$ (as in signal feed and return lines) would generate a nonzero magnetic field near the wires, where the distance from a given point to one wire is noticeably different from the distance to the other wire, but farther away (relative to the wire spacing), where the distances from a given point to either wire are about the same, the fields from both wires tend to cancel out. Thus, maintaining proximity between feed and return paths is an important way to minimize their interference with other signals. The way to maintain their proximity is essentially to minimize their loop area. And, because the mutual inductance from current loop A to current loop B is the same as the mutual inductance from current loop B to current loop A, a circuit that doesn't radiate interference doesn't receive it either.

Thus, from the standpoint of reducing both generation of EMI and susceptibility to EMI, the hard rule is to keep loop areas small. To say that loop areas should be minimized is the same as saying the circuit inductance

should be minimized. Inductance is by definition the constant of proportionality between current and the magnetic field it produces: $\phi = LI$. Holding the feed and return wires close together so as to promote field cancellation can be described either as minimizing the loop area or as minimizing L. It's the same thing.

Shielding

There are three basic kinds of shields: shielding against capacitive coupling, shielding against inductive coupling, and RF shielding. Capacitive coupling is electric field coupling, so shielding against it amounts to shielding against electric fields. As will be seen, this is relatively easy. Inductive coupling is magnetic field coupling, so shielding against it is shielding against magnetic fields. This is a little more difficult. Strangely enough, this type of shielding does not in general involve the use of magnetic materials. RF shielding, the classical "metallic barrier" against all sorts of electromagnetic fields, is what most people picture when they think about shielding. Its effectiveness depends partly on the selection of the shielding material, but mostly, as it turns out, on the treatment of its seams and the geometry of its openings.

SHIELDING AGAINST CAPACITIVE COUPLING

Capacitive coupling involves the passage of interfering signals through mutual or stray capacitances that aren't shown on the circuit diagram, but which the experienced engineer knows are there. Capacitive coupling to one's body is what would cause an unstable oscillator to change its frequency when the person reaches his hand over the circuit, for example. More importantly, in a digital system it causes crosstalk in multi-wire cables.

The way to block capacitive coupling is to enclose the circuit or conductor you want to protect in a metal shield. That's called an electrostatic or Faraday shield. If coverage is 100%, the shield does not have to be grounded, but it usually is, to ensure that circuit-to-shield capacitances go to signal reference ground rather than act as feedback and crosstalk elements. Besides, from a mechanical point of view, grounding it is almost inevitable.

A grounded Faraday shield can be used to break capacitive coupling between a noisy circuit and a victim circuit, as shown in Figure 4. Figure 4a shows two circuits capacitively coupled through the stray capacitance between them. In Figure 4b the stray capacitance is intercepted by a grounded Faraday shield, so that interference currents are shunted to ground. For example, a grounded plane can be inserted between PCBs (printed circuit boards) to eliminate most of the capacitive coupling between them.

Another application of the Faraday shield is in the electrostatically shielded transformer. Here, a conducting foil is laid between the primary and secondary coils so as to intercept the capacitive coupling between them. If a system is being upset by AC line transients, this type of transformer may provide the fix. To be effective in this application, the shield must be connected to the greenwire ground.

SHIELDING AGAINST INDUCTIVE COUPLING

With inductive coupling, the physical mechanism involved is a magnetic flux density B from some external interference source that links with a current loop in the victim circuit, and generates a voltage in the loop in accordance with Lenz's law: $v = -NA(dB/dt)$, where in this case $N = 1$ and A is the area of the current loop in the victim circuit.

There are two aspects to defending a circuit against inductive pickup. One aspect is to try to minimize the offensive fields at their source. This is done by minimizing the area of the current loop at the source so as to promote field cancellation, as described in the section on current loops. The other aspect is to minimize the inductive pickup in the victim circuit by minimizing the area of that current loop, since, from Lenz's law, the induced voltage is proportional to this area. So the two aspects really involve the same corrective action: minimize the areas of the current loops. In other words, minimizing the offensiveness of a circuit inherently minimizes its susceptibility.

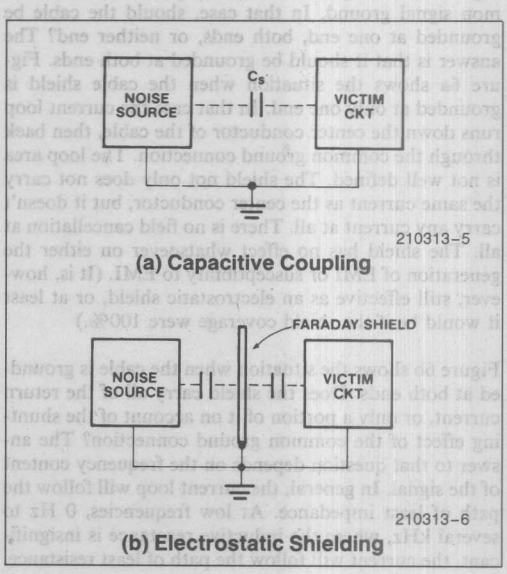


Figure 4. Use of Faraday Shield

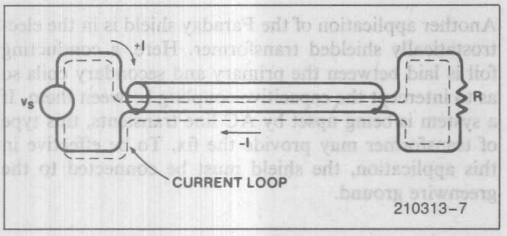


Figure 5. External to the Shield, $\phi = 0$

Shielding against inductive coupling means nothing more nor less than controlling the dimensions of the current loops in the circuit. We must look at four examples of this type of "shielding": the coaxial cable, the twisted pair, the ground plane, and the gridded-ground PCB layout.

The Coaxial Cable—Figure 5 shows a coaxial cable carrying a current I from a signal source to a receiving load. The shield carries the same current as the center conductor. Outside the shield, the magnetic field produced by +I flowing in the center conductor is cancelled by the field produced by −I flowing in the shield. To the extent that the cable is ideal in producing zero external magnetic field, it is immune to inductive pickup from external sources. The cable adds effectively zero area to the loop. This is true only if the shield carries the same current as the center conductor.

In the real world, both the signal source and the receiving load are likely to have one end connected to a common signal ground. In that case, should the cable be grounded at one end, both ends, or neither end? The answer is that it should be grounded at both ends. Figure 6a shows the situation when the cable shield is grounded at only one end. In that case the current loop runs down the center conductor of the cable, then back through the common ground connection. The loop area is not well defined. The shield not only does not carry the same current as the center conductor, but it doesn't carry any current at all. There is no field cancellation at all. The shield has no effect whatsoever on either the generation of EMI or susceptibility to EMI. (It is, however, still effective as an electrostatic shield, or at least it would be if the shield coverage were 100%.)

Figure 6b shows the situation when the cable is grounded at both ends. Does the shield carry all of the return current, or only a portion of it on account of the shunting effect of the common ground connection? The answer to that question depends on the frequency content of the signal. In general, the current loop will follow the path of least impedance. At low frequencies, 0 Hz to several kHz, where the inductive reactance is insignificant, the current will follow the path of least resistance. Above a few kHz, where inductive reactance predominates, the current will follow the path of least inductance. The path of least inductance is the path of

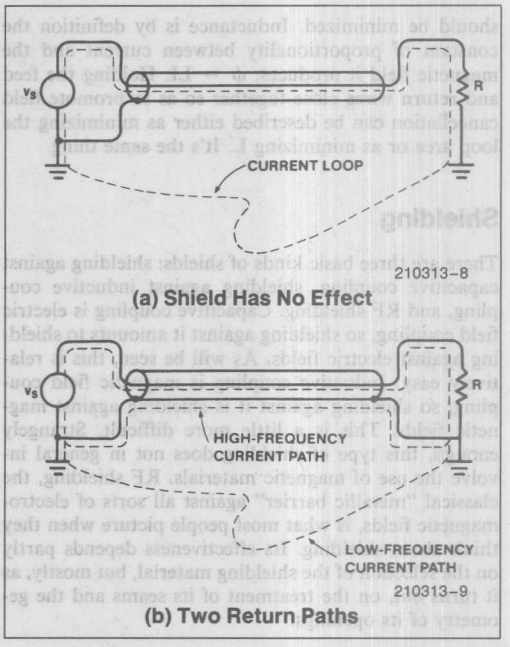


Figure 6. Use of Coaxial Cable

minimum loop area. Hence, for higher frequencies the shield carries virtually the same current as the center conductor, and is therefore effective against both generation and reception of EMI.

Note that we have now introduced the famous "ground loop" problem, as shown in Figure 7a. Fortunately, a digital system has some built-in immunity to moderate ground loop noise. In a noisy environment, however, one can break the ground loop, and still maintain the shielding effectiveness of the coaxial cable, by inserting an optical coupler, as shown in Figure 7b. What the optical coupler does, basically, is allow us to re-define the signal source as being ungrounded, so that that end of the cable need not be grounded, and still lets the shield carry the same current as the center conductor. Obviously, if the signal source weren't grounded in the first place, the optical coupler wouldn't be needed.

The Twisted Pair—A cheaper way to minimize loop area is to run the feed and return wires right next to each other. This isn't as effective as a coaxial cable in minimizing loop area. An ideal coaxial cable adds zero area to the loop, whereas merely keeping the feed and return wires next to each other is bound to add a finite area.

However, two things work to make this cheaper method almost as good as a coaxial cable. First, real coaxial cables are not ideal. If the shield current isn't evenly distributed around the center conductor at every cross-

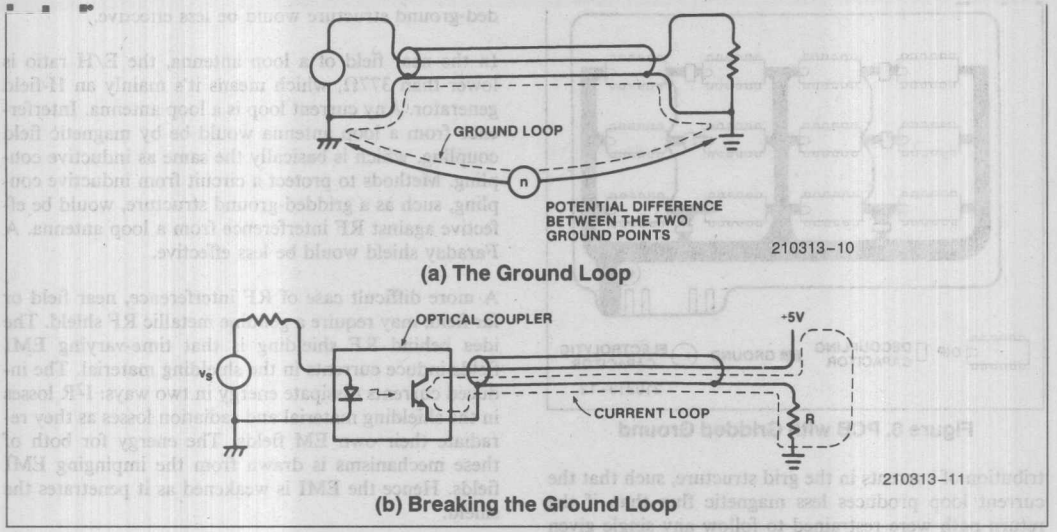


Figure 7. Use of Optical Coupler

section of the cable (it isn't), then field cancellation external to the shield is incomplete. If field cancellation is incomplete, then the effective area added to the loop by the cable isn't zero. Second, in the cheaper method the feed and return wires can be twisted together. This not only maintains their proximity, but the noise picked up in one twist tends to cancel out the noise picked up in the next twist down the line. Thus the "twisted pair" turns out to be about as good a shield against inductive coupling as coaxial cable is.

The twisted pair does not, however, provide electrostatic shielding (i.e., shielding against capacitive coupling). Another operational difference between them is that the coaxial cable works better at higher frequencies. This is primarily because the twisted pair adds more capacitive loading to the signal source than the coaxial cable does. The twisted pair is normally considered useful up to only about 1 MHz, as opposed to near a GHz for the coaxial cable.

The Ground Plane—The best way to minimize loop areas when many current loops are involved is to use a ground plane. A ground plane is a conducting surface that is to serve as a return conductor for all the current loops in the circuit. Normally, it would be one or more layers of a multilayer PCB. All ground points in the circuit go not to a grounded trace on the PCB, but directly to the ground plane. This leaves each current loop in the circuit free to complete itself in whatever configuration yields minimum loop area (for frequencies wherein the ground path impedance is primarily inductive).

Thus, if the feed path for a given signal zigzags its way across the PCB, the return path for this signal is free to zigzag right along beneath it on the ground plane, in such a configuration as to minimize the energy stored in the magnetic field produced by this current loop. Minimal magnetic flux means minimal effective loop area and minimal susceptibility to inductive coupling.

The Gridded-Ground PCB Layout—The next best thing to a ground plane is to lay out the ground traces on a PCB in the form of a grid structure, as shown in Figure 8. Laying horizontal traces on one side of the board and vertical traces on the other side allows the passage of signal and power traces. Wherever vertical and horizontal ground traces cross, they must be connected by a feed-through.

Have we not created here a network of "ground loops"? Yes, in the literal sense of the word, but loops in the ground layout on a PCB are not to be feared. Such inoffensive little loops have never caused as much noise pickup as their avoidance has. Trying to avoid innocent little loops in the ground layout, PCB designers have forced current loops into geometries that could swallow a whale. That is exactly the wrong thing to do.

The gridded ground structure works almost as well as the ground plane, as far as minimizing loop area is concerned. For a given current loop, the primary return path may have to zig once in a while where its feed path zags, but you still get a mathematically optimal dis-

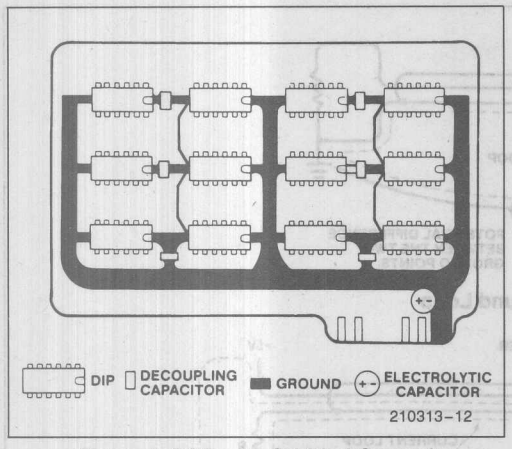


Figure 8. PCB with Gridded Ground

tribution of currents in the grid structure, such that the current loop produces less magnetic flux than if the return path were restrained to follow any single given ground trace. The key to attaining minimum loop areas for all the current loops together is to let the ground currents distribute themselves around the entire area of the board as freely as possible. They want to minimize their own magnetic field. Just let them.

RF SHIELDING

A time-varying electric field generates a time-varying magnetic field, and vice versa. Far from the source of a time-varying EM field, the ratio of the amplitudes of the electric and magnetic fields is always 377Ω. Up close to the source of the fields, however, this ratio can be quite different, and dependent on the nature of the source. Where the ratio is near 377Ω is called the far field, and where the ratio is significantly different from 377Ω is called the near field. The ratio itself is called the wave impedance, E/H.

The near field goes out about 1/6 of a wavelength from the source. At 1 MHz this is about 150 feet, and at 10 MHz it's about 15 feet. That means if an EMI source is in the same room with the victim circuit, it's likely to be a near field problem. The reason this matters is that in the near field an RF interference problem could be almost entirely due to E-field coupling or H-field coupling, and that could influence the choice of an RF shield or whether an RF shield will help at all.

In the near field of a whip antenna, the E/H ratio is higher than 377Ω, which means it's mainly an E-field generator. A wire-wrap post can be a whip antenna. Interference from a whip antenna would be by electric field coupling, which is basically capacitive coupling. Methods to protect a circuit from capacitive coupling, such as a Faraday shield, would be effective against RF interference from a whip antenna. A gridded-ground structure would be less effective.

In the near field of a loop antenna, the E/H ratio is lower than 377Ω, which means it's mainly an H-field generator. Any current loop is a loop antenna. Interference from a loop antenna would be by magnetic field coupling, which is basically the same as inductive coupling. Methods to protect a circuit from inductive coupling, such as a gridded-ground structure, would be effective against RF interference from a loop antenna. A Faraday shield would be less effective.

A more difficult case of RF interference, near field or far field, may require a genuine metallic RF shield. The idea behind RF shielding is that time-varying EMI fields induce currents in the shielding material. The induced currents dissipate energy in two ways: I^2R losses in the shielding material and radiation losses as they re-radiate their own EM fields. The energy for both of these mechanisms is drawn from the impinging EMI fields. Hence the EMI is weakened as it penetrates the shield.

More formally, the I^2R losses are referred to as absorption loss, and the re-radiation is called reflection loss. As it turns out, absorption loss is the primary shielding mechanism for H-fields, and reflection loss is the primary shielding mechanism for E-fields. Reflection loss, being a surface phenomenon, is pretty much independent of the thickness of the shielding material. Both loss mechanisms, however, are dependent on the frequency (ω) of the impinging EMI field, and on the permeability (μ) and conductivity (σ) of the shielding material. These loss mechanisms vary approximately as follows:

$$\text{reflection loss to an E-field (in dB)} \sim \log \frac{\sigma}{\omega\mu}$$

$$\text{absorption loss to an H-field (in dB)} \sim t\sqrt{\omega\sigma\mu}$$

where t is the thickness of the shielding material.

The first expression indicates that E-field shielding is more effective if the shield material is highly conductive, and less effective if the shield if ferromagnetic, and that low-frequency fields are easier to block than high-frequency fields. This is shown in Figure 9.

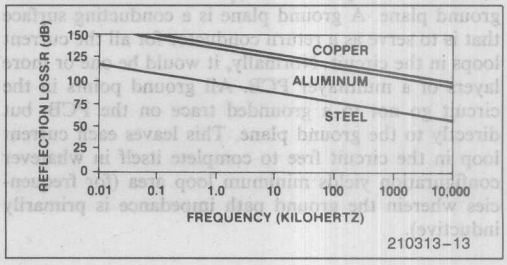


Figure 9. E-Field Shielding

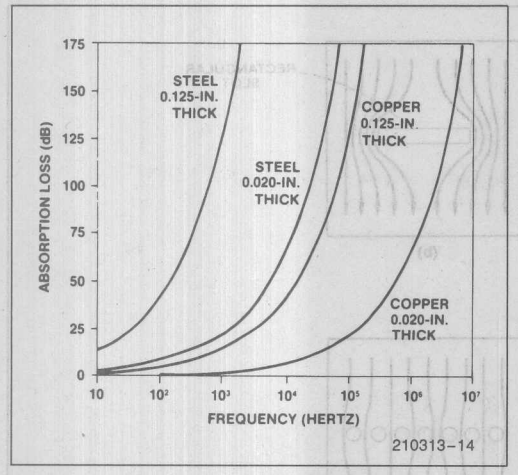


Figure 10. H-Field Shielding

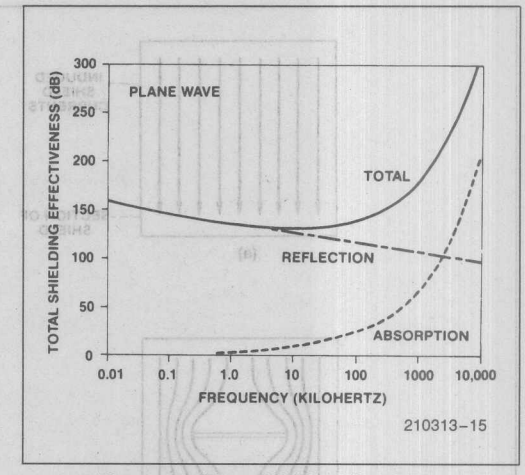


Figure 11. E- and H-Field Shielding

Copper and aluminum both have the same permeability, but copper is slightly more conductive, and so provides slightly greater reflection loss to an E-field. Steel is less effective for two reasons. First, it has a somewhat elevated permeability due to its iron content, and second, as tends to be the case with magnetic materials, it is less conductive.

On the other hand, according to the expression for absorption loss to an H-field, H-field shielding is more effective at higher frequencies and with shield material that has both high conductivity and high permeability. In practice, however, selecting steel for its high permeability involves some compromise in conductivity. But the increase in permeability more than makes up for the decrease in conductivity, as can be seen in Figure 10. This figure also shows the effect of shield thickness.

A composite of E-field and H-field shielding is shown in Figure 11. However, this type of data is meaningful only in the far field. In the near field the EMI could be 90% H-field, in which case the reflection loss is irrelevant. It would be advisable then to beef up the absorption loss, at the expense of reflection loss, by choosing steel. A better conductor than steel might be less expensive, but quite ineffective.

A different shielding mechanism that can be taken advantage of for low frequency magnetic fields is the ability of a high permeability material such as mumetal to divert the field by presenting a very low reluctance path to the magnetic flux. Above a few kHz, however, the permeability of such materials is the same as steel.

In actual fact the selection of a shielding material turns out to be less important than the presence of seams, joints and holes in the physical structure of the enclosure. The shielding mechanisms are related to the induction of currents in the shield material, but the currents must be allowed to flow freely. If they have to detour around slots and holes, as shown in Figure 12, the shield loses much of its effectiveness.

As can be seen in Figure 12, the severity of the detour has less to do with the area of the hole than it does with the geometry of the hole. Comparing Figure 12c with 12d shows that a long narrow discontinuity such as a seam can cause more RF leakage than a line of holes with larger total area. A person who is responsible for designing or selecting rack or chassis enclosures for an EMI environment needs to be familiar with the techniques that are available for maintaining electrical continuity across seams. Information on these techniques is available in the references.

Grounds

There are two kinds of grounds: earth-ground and signal ground. The earth is not an equipotential surface, so earth ground potential varies. That and its other electrical properties are not conducive to its use as a return conductor in a circuit. However, circuits are often connected to earth ground for protection against shock hazards. The other kind of ground, signal ground, is an arbitrarily selected reference node in a circuit—the node with respect to which other node voltages in the circuit are measured.

SAFETY GROUND

The standard 3-wire single-phase AC power distribution system is represented in Figure 13. The white wire is earth-grounded at the service entrance. If a load circuit has a metal enclosure or chassis, and if the black wire develops a short to the enclosure, there will be a shock hazard to operating personnel, unless the enclosure itself is earth-grounded. If the enclosure is earth-

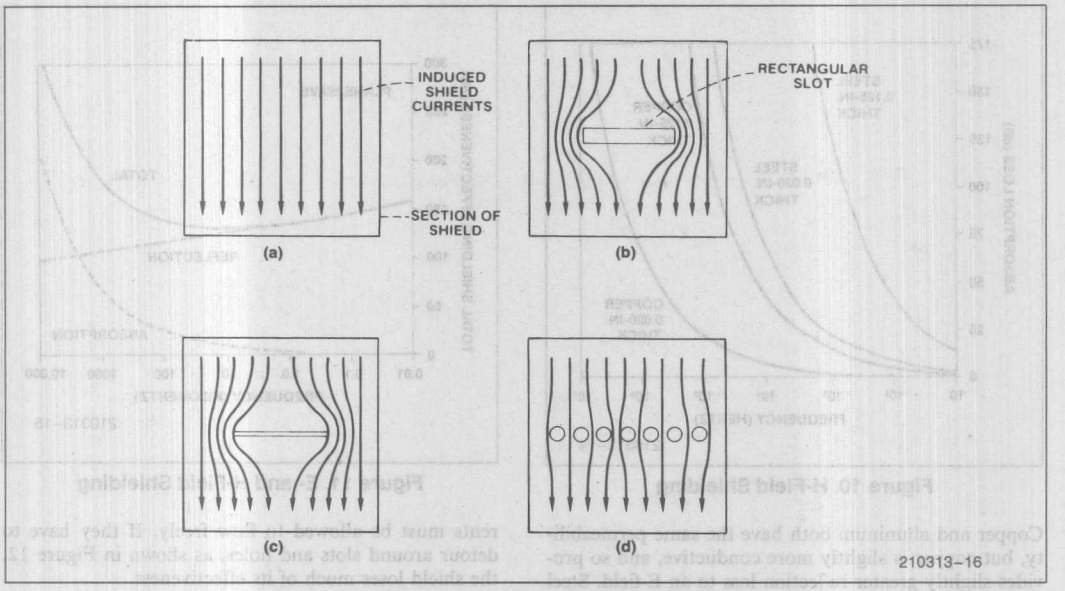


Figure 12. Effect of Shield Discontinuity on Magnetically Induced Shield Current

grounded, a short results in a blown fuse rather than a "hot" enclosure. The earth-ground connection to the enclosure is called a safety ground. The advantage of the 3-wire power system is that it distributes a safety ground along with the power.

Note that the safety-ground wire carries no current, except in case of a fault, so that at least for low frequencies it's at earth-ground potential along its entire length. The white wire, on the other hand, may be several volts off ground, due to the IR drop along its length.

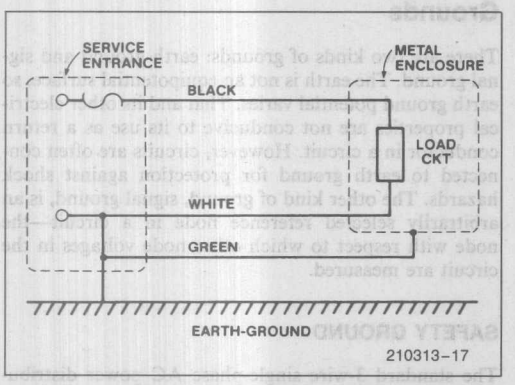


Figure 13. Single-Phase Power Distribution

SIGNAL GROUND

Signal ground is a single point in a circuit that is designated to be the reference node for the circuit. Commonly, wires that connect to this single point are also referred to as "signal ground." In some circles "power supply common" or PSC is the preferred terminology for these conductors. In any case, the manner in which these wires connect to the actual reference point is the basis of distinction among three kinds of signal-ground wiring methods: series, parallel, and multipoint. These methods are shown in Figure 14.

The series connection is pretty common because it's simple and economical. It's the noisiest of the three, however, due to common ground impedance coupling between the circuits. When several circuits share a ground wire, currents from one circuit, flowing through the finite impedance of the common ground line, cause variations in the ground potential of the other circuits. Given that the currents in a digital system tend to be spiked, and that the common impedance is mainly inductive reactance, the variations could be bad enough to cause bit errors in high current or particularly noisy situations.

The parallel connection eliminates common ground impedance problems, but uses a lot of wire. Other disadvantages are that the impedance of the individual ground lines can be very high, and the ground lines themselves can become sources of EMI.

nized by using a ground plane with the various circuits connected to it by very short ground leads. This type of connection would be used mainly in RF circuits above 10 MHz.

PRACTICAL GROUNDING

A combination of series and parallel ground-wiring methods can be used to trade off economic and the various electrical considerations. The idea is to run series connections for circuits that have similar noise properties, and connect them at a single reference point, as in the parallel method, as shown in Figure 15.

In Figure 15, "noisy signal ground" connects to things like motors and relays. Hardware ground is the safety ground connection to chassis, racks, and cabinets. It's a mistake to use the hardware ground as a return path for signal currents because it's fairly noisy (for example, it's the hardware ground that receives an ESD spark) and tends to have high resistance, due to joints and seams.

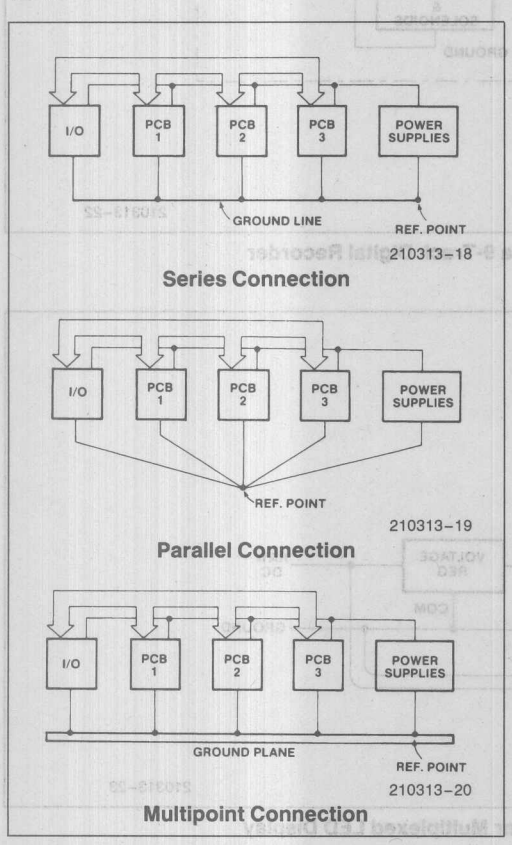


Figure 14. Three Ways to Wire the Grounds

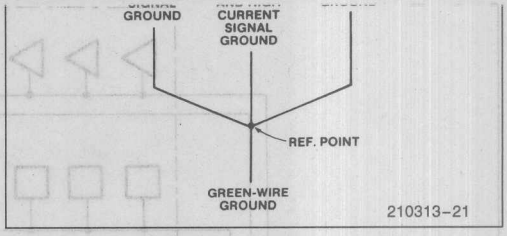


Figure 15. Parallel Connection of Series Grounds

Screws and bolts don't always make good electrical connections because of galvanic action, corrosion, and dirt. These kinds of connections may work well at first, and then cause mysterious maladies as the system ages.

Figure 16 illustrates a grounding system for a 9-track digital tape recorder, showing an application of the series/parallel ground-wiring method.

Figure 17 shows a similar separation of grounds at the PCB level. Currents in multiplexed LED displays tend to put a lot of noise on the ground and supply lines because of the constant switching and changing involved in the scanning process. The segment driver ground is relatively quiet, since it doesn't conduct the LED currents. The digit driver ground is noisier, and should be provided with a separate path to the PCB ground terminal, even if the PCB ground layout is gridded. The LED feed and return current paths should be laid out on opposite sides of the board like parallel flat conductors.

Figure 18 shows right and wrong ways to make ground connections in racks. Note that the safety ground connections from panel to rack are made through ground straps, not panel screws. Rack 1 correctly connects signal ground to rack ground only at the single reference point. Rack 2 incorrectly connects signal ground to rack ground at two points, creating a ground loop around points 1, 2, 3, 4, 1.

Breaking the "electronics ground" connection to point 1 eliminates the ground loop, but leaves signal ground in rack 2 sharing a ground impedance with the relatively noisy hardware ground to the reference point; in fact, it may end up using hardware ground as a return path for signal and power supply currents. This will probably cause more problems than the ground loop.

BRAIDED CABLE

Ground impedance problems can be virtually eliminated by using braided cable. The reduction in impedance is due to skin effect: At higher frequencies the current tends to flow along the surface of a conductor rather

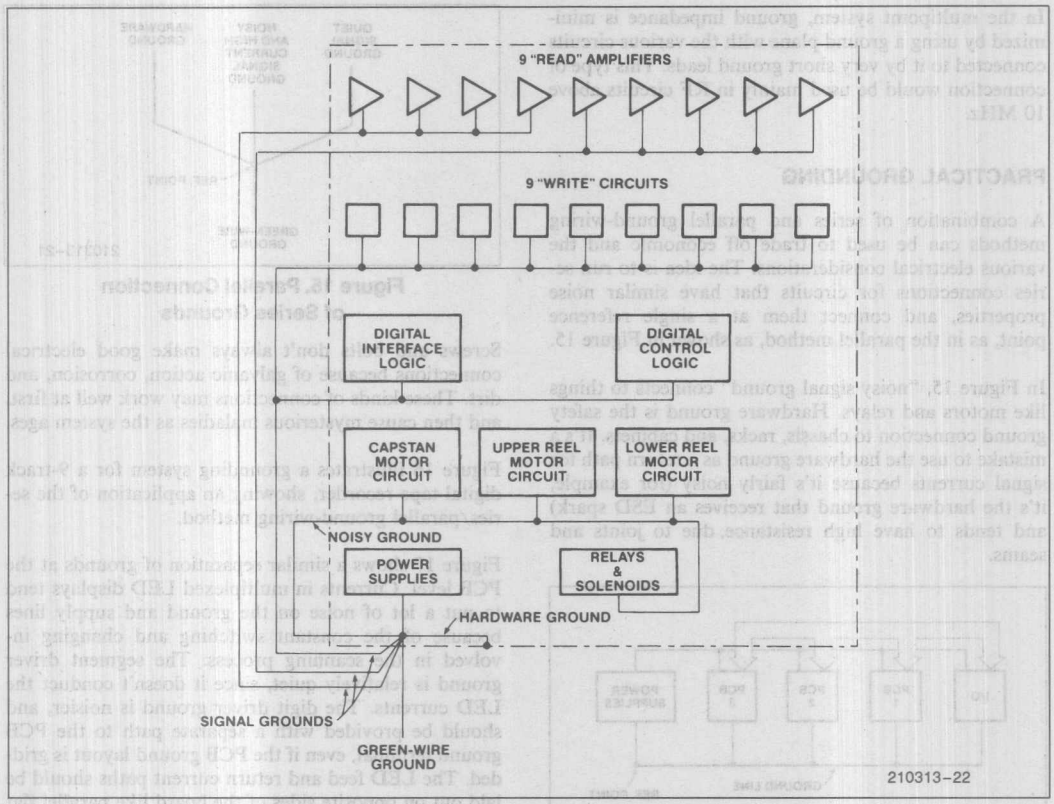


Figure 16. Ground System in a 9-Track Digital Recorder

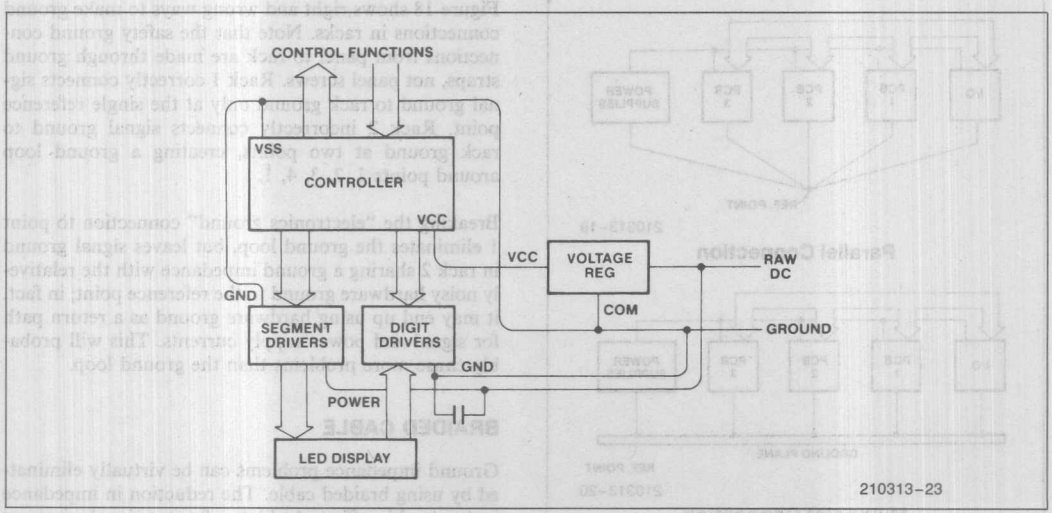


Figure 17. Separate Ground for Multiplexed LED Display

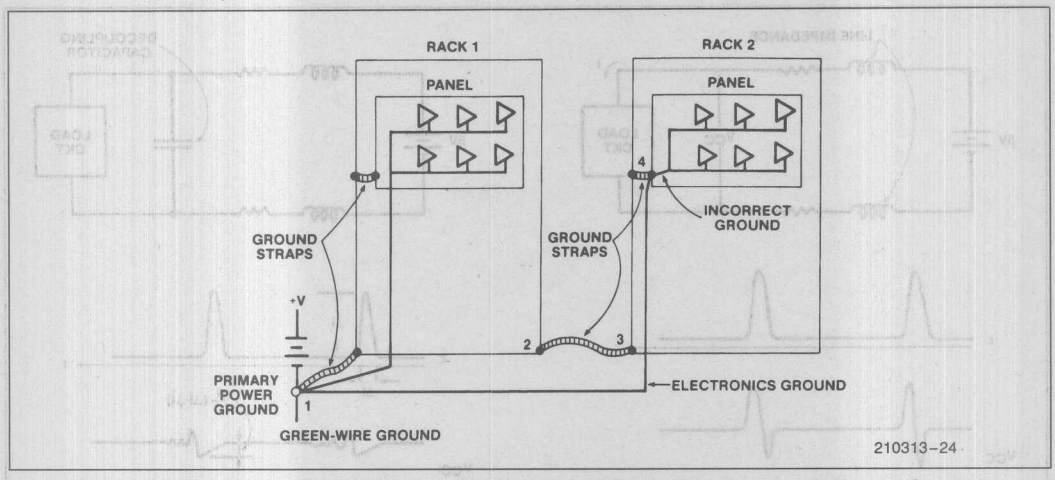


Figure 18. Electronic Circuits Mounted in Equipment Racks Should Have Separate Ground Connections. Rack 1 Shows Correct Grounding, Rack 2 Shows Incorrect Grounding.

than uniformly through its bulk. While this effect tends to increase the impedance of a given conductor, it also indicates the way to minimize impedance, and that is to manipulate the shape of the cross-section so as to provide more surface area. For its bulk, braided cable is almost pure surface.

Power Supply Distribution and Decoupling

The main consideration for power supply distribution lines is, as for signal lines, to minimize the areas of the current loops. But the power supply lines take on an importance that no signal line has when one considers the fact that these lines have access to every PC board in the system. The very extensiveness of the supply current loops makes it difficult to keep loop areas small. And, a noise glitch on a supply line is a glitch delivered to every board in the system.

The power supply provides low-frequency current to the load, but the inductance of the board-to-board and chip-to-chip distribution network makes it difficult for the power supply to maintain VCC specs on the chip while providing the current spikes that a digital system requires. In addition, the power supply current loop is a very large one, which means there will be a lot of noise pick-up. Figure 19a shows a load circuit trying to draw current spikes from a supply voltage through the line impedance. To the VCC waveform shown in that figure should be added the inductive pick-up associated with a large loop area.

Adding a decoupling capacitor solves two problems: The capacitor acts as a nearby source of charge to supply the current spikes through a smaller line impedance, and it defines a much smaller loop area for the higher frequency components of EMI. This is illustrated in Figure 19b, which shows the capacitor supplying the current spike, during which VCC drops from 5V by the amount indicated in the figure. Between current spikes the capacitor recovers through the line impedance.

One should resist the temptation to add a resistor or an inductor to the decoupler so as to form a genuine RC or LC low-pass filter because that slows down the speed with which the decoupler cap can be refreshed. Good filtering and good decoupling are not necessarily the same thing.

The current loop for the higher frequency currents, then, is defined by the decoupling cap and the load circuit, rather than by the power supply and the load circuit. For the decoupling cap to be able to provide the current spikes required by the load, the inductance of this current loop must be kept small, which is the same as saying the loop area must be kept small. This is also the requirement for minimizing inductive pick-up in the loop.

There are two kinds of decoupling caps: board decouplers and chip decouplers. A board decoupler will normally be a 10 to 100 μF electrolytic capacitor placed near to where the power supply enters the PC board, but its placement is relatively non-critical. The purpose of the board decoupler is to refresh the charge on the chip decouplers. The chip decouplers are what actually provide the current spikes to the chips. A chip decoupler will normally be a 0.1 to 1 μF ceramic capacitor placed near the chip and connected to the chip by traces that minimize the area of the loop formed by the cap and the chip. If a chip decoupler is not properly placed on the board, it will be ineffective as a decoupler

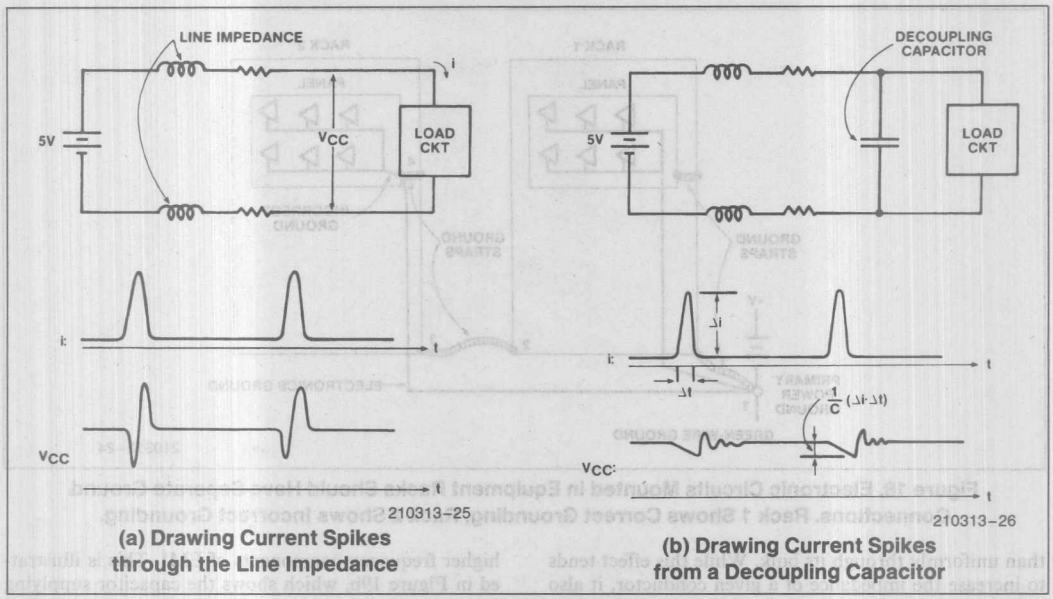


Figure 19. What a Decoupling Capacitor Does

and will serve only to increase the cost of the board. Good and bad placement of decoupling capacitors are illustrated in Figure 20.

Power distribution traces on the PC board need to be laid out so as to obtain minimal area (minimal inductance) in the loops formed by each chip and its decoupler, and by the chip decouplers and the board decoupler. One way to accomplish this goal is to use a power plane. A power plane is the same as a ground plane, but at VCC potential. More economically, a power grid similar to the ground grid previously discussed (Figure 8) can be used. Actually, if the chip decoupling loops are small, other aspects of the power layout are less critical. In other words, power planes and power gridding aren't needed, but power traces *should* be laid in the closest possible proximity to ground traces, preferably so that each power trace is on the direct opposite side of the board from a ground trace.

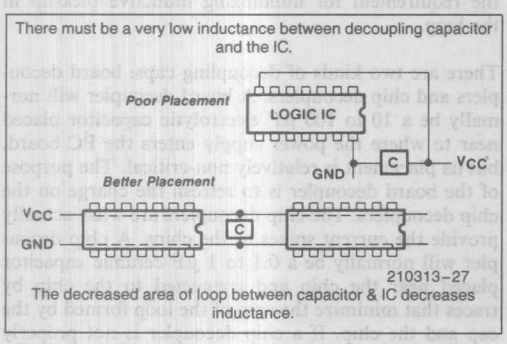


Figure 20. Placement of Decoupling Capacitors

Special-purpose power supply distribution buses which mount on the PCB are available. The buses use a parallel flat conductor configuration, one conductor being a VCC line and the other a ground line. Used in conjunction with a gridded ground layout, they not only provide a low-inductance distribution system, but can themselves form part of the ground grid, thus facilitating the PCB layout. The buses are available with and without enhanced bus capacitance, under the names Mini/Bus® and Q/PAC® from Rogers Corp. (5750 E. McKellips, Mesa, AZ 85205).

SELECTING THE VALUE OF THE DECOUPLING CAP

The effectiveness of the decoupling cap has a lot to do with the way the power and ground traces connect this capacitor to the chip. In fact, the area formed by this loop is more important than the value of the capacitance. Then, given that the area of this loop is indeed minimal, it can generally be said that the larger the value of the decoupling cap, the more effective it is, if the cap has a mica, ceramic, glass, or polystyrene dielectric.

It's often said, and not altogether accurately, that the chip decoupler shouldn't have too large a value. There are two reasons for this statement. One is that some capacitors, because of the nature of their dielectrics, tend to become inductive or lossy at higher frequencies. This is true of electrolytic capacitors, but mica, glass,

al hundred MHz. The other reason cited for not using too large a capacitance has to do with lead inductance.

The capacitor with its lead inductance forms a series LC circuit. Below the frequency of series resonance, the net impedance of the combination is capacitive. Above that frequency, the net impedance is inductive. Thus a decoupling capacitor is capacitive only below the frequency of series resonance. The frequency is given by

$$f_0 = \frac{1}{2\pi\sqrt{LC}}$$

where C is the decoupling capacitance and L is the lead inductance between the capacitor and the chip. On a PC board this inductance is determined by the layout, and is the same whether the capacitor dropped into the PCB holes is 0.001 μF or 1 μF. Thus, increasing the capacitance lowers the series resonant frequency. In fact, according to the resonant frequency formula, increasing C by a factor of 100 lowers the resonant frequency by a factor of 10.

Figures quoted on the series resonant frequency of a 0.01 μF capacitor run from 10 to 15 MHz, depending on the lead length. If these numbers were accurate, a 1 μF capacitor in the same position on the board would have a resonant frequency of 1.0 to 1.5 MHz, and as a decoupler would do more harm than good. However, the numbers are based on a presumed inductance of a given length of wire (the lead length). It should be noted that a "length of wire" has no inductance at all, strictly speaking. Only a complete current loop has inductance, and the inductance depends on the geometry of the loop. Figures quoted on the inductance of a length of wire are based on a presumably "very large" loop area, such that the magnetic field produced by the return current has no cancellation effect on the field produced by the current in the given length of wire. Such a loop geometry is not and should not be the case with the decoupling loop.

Figure 21 shows VCC waveforms, measured between pins 40 and 20 (VCC and VSS) of an 8751 CPU, for several conditions of decoupling on a PC board that has a decoupling loop area slightly larger than necessary. These photographs show the effects of increasing the decoupling capacitance and decreasing the area of the decoupling loop. The indications are that a 1 μF capacitor is better than a 0.1 μF capacitor, which in turn is better than nothing, and that the board should have been laid out with more attention paid to the area of the decoupling loop.

Figure 21e was obtained using a special-purpose experimental capacitor designed by Rogers Corp. (Q-Pac Division, Mesa, AZ) for use as a decoupler. It consists of two parallel plates, the length of a 40-pin DIP, separated by a ceramic dielectric. Sandwiched between the and the PCB), it makes connection to pins 40 and 20, forming a leadless decoupling capacitor. It is obviously a configuration of minimal inductance. Unfortunately, the particular sample tested had only 0.07 μF of capacitance and so was unable to prevent the 1 MHz ripple as effectively as the configuration of Figure 21d. It seems apparent, though, that with more capacitance this part will alleviate a lot of decoupling problems.

THE CASE FOR ON-BOARD VOLTAGE REGULATION

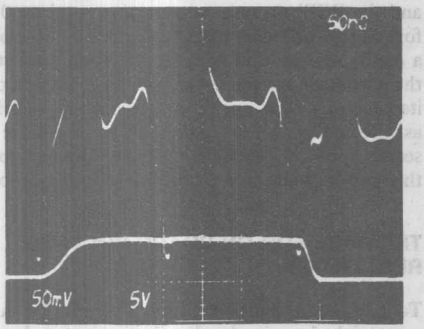
To complicate matters, supply line glitches aren't always picked up in the distribution networks, but can come from the power supply circuit itself. In that case, a well-designed distribution network faithfully delivers the glitch throughout the system. The VCC glitch in Figure 22 was found to be coming from within a bench power supply in response to the EMP produced by an induction coil spark generator that was being used at Intel during a study of noise sensitivity. The VCC glitch is about 400 mV high and some 20 μs in duration. Normal board decoupling techniques were ineffective in removing it, but adding an on-board voltage regulator chip did the job.

Thus, a good case can be made in favor of using a voltage regulator chip on each PCB, instead of doing all the voltage regulation at the supply circuit. This eases requirements on the heat-sinking at the supply circuit, and alleviates much of the distribution and board decoupling headaches. However, it also brings in the possibility that different boards would be operating at slightly different VCC levels due to tolerance in the regulator chips; this then leads to slightly different logic levels from board to board. The implications of that may vary from nothing to latch-up, depending on what kinds of chips are on the boards, and how they react to an input "high" that is perhaps 0.4V higher than local VCC.

Recovering Gracefully from a Software Upset

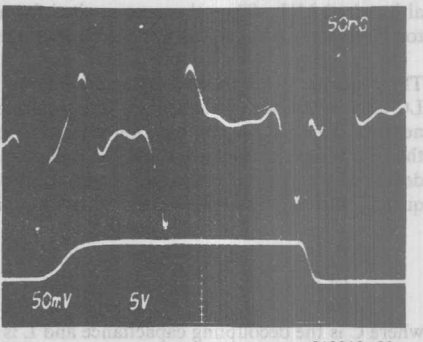
Even when one follows all the best guidelines for designing for a noisy environment, it's always possible for a noise transient to occur which exceeds the circuit's immunity level. In that case, one can strive at least for a graceful recovery.

Graceful recovery schemes involve additional hardware and/or software which is supposed to return the system to a normal operating mode after a software upset has occurred. Two decisions have to be made: How to recognize when an upset has occurred, and what to do about it.



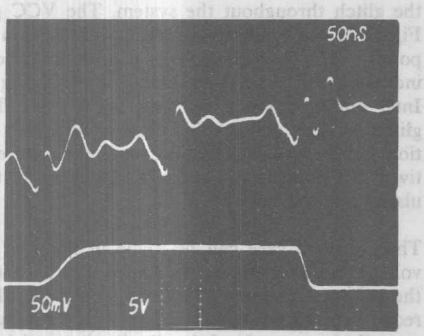
210313-28

(a) No Decoupling Cap



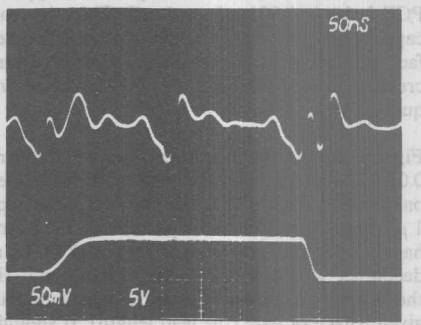
210313-29

(b) 0.1 μF Decoupler in Place on the PCB



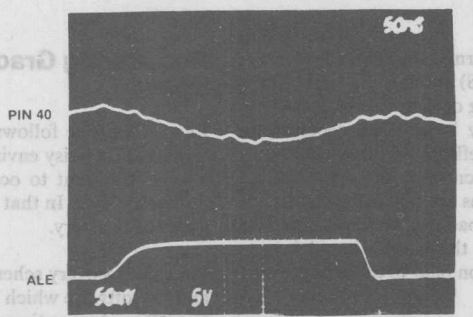
210313-30

(c) 0.1 μF Decoupler Stretched Directly from Pin 40 to Pin 20, under the Socket. (The difference between this and 21b is due only to the change in loop geometry. Also shown is the upward slope of a ripple in VCC. The ripple frequency is 1 MHz, the same as ALE.)



210313-31

(d) 1.0 μF Decoupler Stretched Directly from Pin 40 to Pin 20, under the Socket. (This prevents the 1 MHz ripple, but there's no reduction in higher frequency components. Further increases in capacitance effected no further improvement.)



210313-32

(e) Special-Purpose Decoupling Cap under Development by Rogers Corp. (Further discussion in text.)

Figure 21. Noise on VCC Line

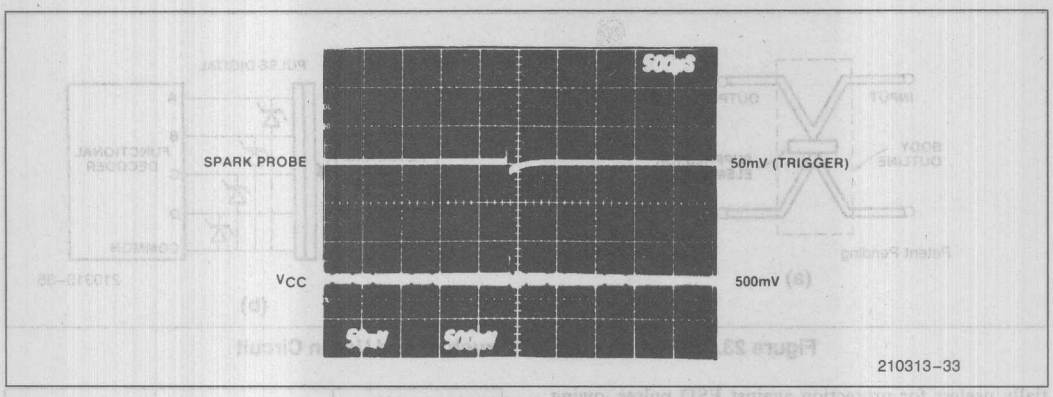


Figure 22. EMP-Induced Glitch

If the designer knows what kinds and combinations of outputs can legally be generated by the system, he can use gates to recognize and flag the occurrence of an illegal state of affairs. The flag can then trigger a jump to a recovery routine which then may check or re-initialize data, perhaps output an error message, or generate a simple reset.

The most reliable scheme is to use a so-called watchdog circuit. Here the CPU is programmed to generate a periodic signal as long as the system is executing instructions in an expected manner. The periodic signal is then used to hold off a circuit that will trigger a jump to a recovery routine. The periodic signal needs to be AC-coupled to the trigger circuit so that a "stuck-at" fault won't continue to hold off the trigger. Then, if the processor locks up someplace, the periodic signal is lost and the watchdog triggers a reset.

In practice, it may be convenient to drive the watchdog circuit with a signal which is being generated anyway by the system. One needs to be careful, however, that an upset does in fact discontinue that signal. Specifically, for example, one could use one of the digit drive signals going to a multiplexed display. But display scanning is often handled in response to a timer-interrupt, which may continue operating even though the main program is in a failure mode. Even so, with a little extra software, the signal can be used to control the watchdog (see Reference 8 on this).

Simpler schemes can work well for simpler systems. For example, if a CPU isn't doing anything but scanning and decoding a keyboard, there's little to lose and much to gain by simply resetting it periodically with an astable multivibrator. It only takes about 13 μs (at 6 MHz) to reset an 8048 if the clock oscillator is already running.

A zero-cost measure is simply to fill all unused program memory with NOPs and JMPs to a recovery routine. The effectiveness of this method is increased by writing the program in segments that are separated by NOPs and JMPs. It's still possible, of course, to get hung up in a data table or something. But you get a lot of protection, for the cost.

Further discussion of graceful recovery schemes can be found in Reference 13.

Special Problem Areas

ESD

MOS chips have some built-in protection against a static charge build-up on the pins, as would occur during normal handling, but there's no protection against the kinds of current levels and rise times that occur in a genuine electrostatic spark. These kinds of discharges can blow a crater in the silicon.

It must be recognized that connecting CPU pins unprotected to a keyboard or to anything else that is subject to electrostatic discharges makes an extremely fragile configuration. Buffering them is the very least one can do. But buffering doesn't completely solve the problem, because then the buffer chips will sustain the damage (even TTL); therefore, one might consider mounting the buffer chips in sockets for ease of replacement.

Transient suppressors, such as the TranZorbs® made by General Semiconductor Industries (Tempe, AZ), may in the long run provide the cheapest protection if their "zero inductance" structure is used. The structure and circuit application are shown in Figure 23.

The suppressor element is a pn junction that operates like a Zener diode. Back-to-back units are available for AC operation. The element is more or less an open circuit at normal system voltage (the standoff voltage rating for the device), and conducts like a Zener diode at the clamping voltage.

The lead inductance in the conventional transient suppressor package makes the conventional package essen-

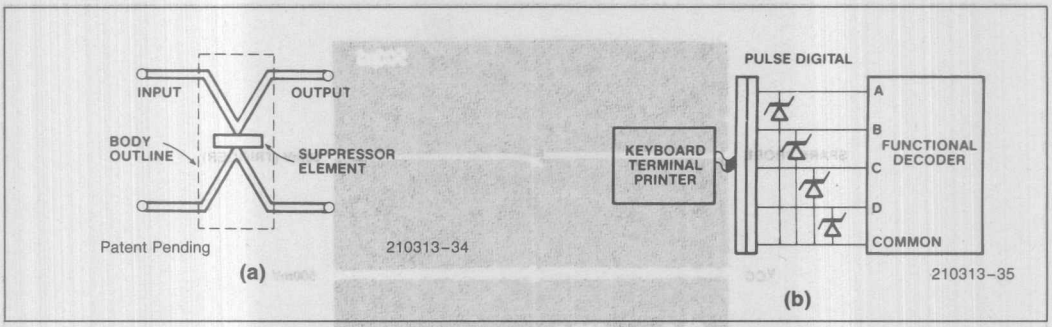


Figure 23. "Zero-Inductance" Structure and Use in Circuit

tially useless for protection against ESD pulses, owing to the fast rise of these pulses. The "zero inductance" units are available singly in a 4-pin DIP, and in arrays of four to a 16-pin DIP for PCB level protection. In that application they should be mounted in close proximity to the chips they protect.

In addition, metal enclosures or frames or parts that can receive an ESD spark should be connected by braided cable to the green-wire ground. Because of the ground impedance, ESD current shouldn't be allowed to flow through any signal ground, even if the chips are protected by transient suppressors. A 35 kV ESD spark can always spare a few hundred volts to drive a fast current pulse down a signal ground line if it can't find a braided cable to follow. Think how delighted your 8048 will be to find its VSS pin 250V higher than VCC for a few 10s of nanoseconds.

THE AUTOMOTIVE ENVIRONMENT

The automobile presents an extremely hostile environment for electronic systems. There are several parts to it:

1. Temperature extremes from −40°C to +125°C (under the hood) or +85°C (in the passenger compartment)
2. Electromagnetic pulses from the ignition system
3. Supply line transients that will knock your socks off

One needs to take a long, careful look at the temperature extremes. The allowable storage temperature range for most Intel MOS chips is −65°C to +150°C, although some chips have a maximum storage temperature rating of +125°C. In operation (or "under bias," as the data sheets say) the allowable ambient temperature range depends on the product grade, as follows:

Grade	Ambient Temperature	
	Min	Max
Commercial	0	70
Industrial	−40	+85
Automotive	−40	+110
Military	−55	+125

The different product grades are actually the same chip, but tested according to different standards. Thus, a given commercial-grade chip might actually pass military temperature requirements, but not have been tested for it. (Of course, there are other differences in grading requirements having to do with packaging, burn-in, traceability, etc.)

In any case, it's apparent that commercial-grade chips can't be used safely in automotive applications, not even in the passenger compartment. Industrial-grade chips can be used in the passenger compartment, and automotive or military chips are required in under-the-hood applications.

Ignition noise, CB radios, and that sort of thing are probably the least of your worries. In a poorly designed system, or in one that has not been adequately tested for the automotive environment, this type of EMI might cause a few software upsets, but not destroy chips.

The major problem, and the one that seems to come as the biggest surprise to most people, is the line transients. Regrettably, the 12V battery is not actually the source of power when the car is running. The charging system is, and it's not very clean. The only time the battery is the real source of power is when the car is first being started, and in that condition the battery terminals may be delivering about 5V or 6V. As follows is a brief description of the major idiosyncracies of the "12V" automotive power line.

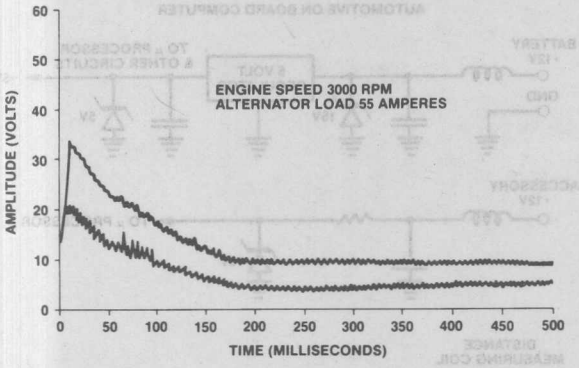


Figure 24. Typical Load Dump Transients

- An abrupt reduction in the alternator load causes a positive voltage transient called "load dump." In a load dump transient the line voltage rises to 20V or 30V in a few μs, then decays exponentially with a time constant of about 100 μs, as shown in Figure 24. Much higher peak voltages and longer decay times have also been reported. The worst case load dump is caused by disconnecting a low battery from the alternator circuit while the alternator is running. Normally this would happen intermittently when the battery terminal connections are defective.
- When the ignition is turned off, as the field excitation decays, the line voltage can go to between -40V and -100V for 100 μs or more.
- Miscellaneous solenoid switching transients, such as the one shown in Figure 25, can drive the line to + or -200V to 400V for several μs.
- Mutual coupling between unshielded wires in long harnesses can induce 100V and 200V transients in unprotected circuits.

What all this adds up to is that people in the business of building systems for automotive applications need a comprehensive testing program. An SAE guideline which describes the automotive environment is available to designers: SAE J1211, "Recommended Environmental Practices for Electronic Equipment Design," *1980 SAE Handbook,* Part 1, pp. 22.80–22.96.

Some suggestions for protecting circuitry are shown in Figure 26. A transient suppressor is placed in front of the regulator chip to protect it. Since the rise times in these transients are not like those in ESD pulses, lead inductance is less critical and conventional devices can be used. The regulator itself is pretty much of a necessity, since a load dump transient is simply not going to be removed by any conventional LC or RC filter.

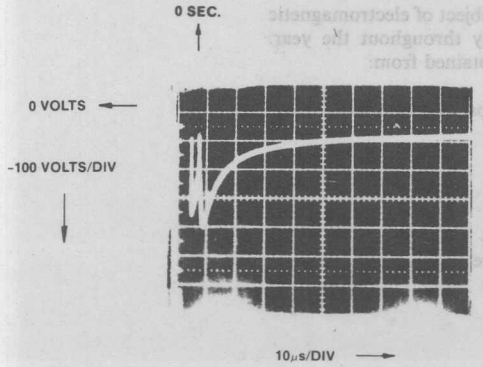


Figure 25. Transient Created by De-energizing an Air Conditioning Clutch Solenoid

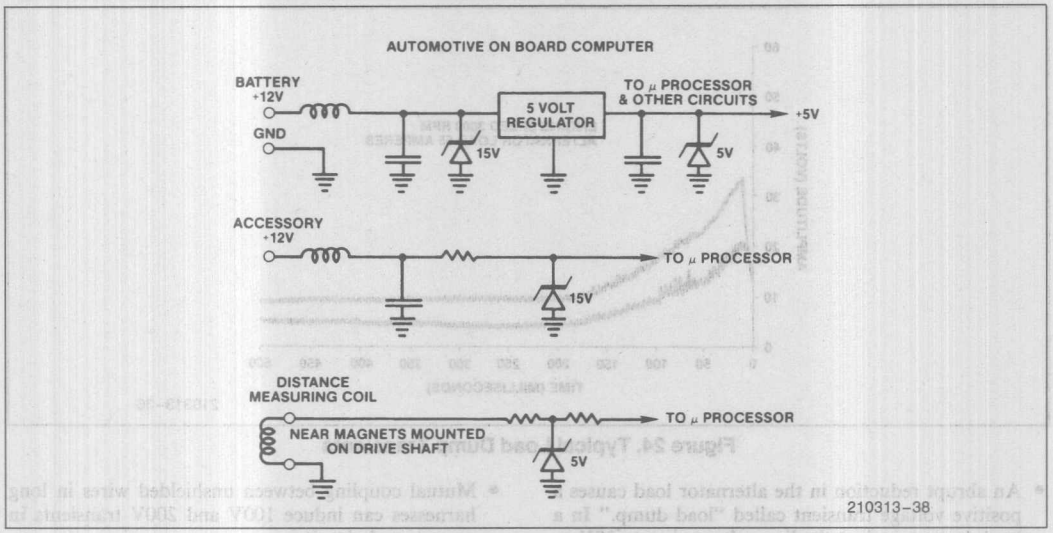


Figure 26. Use of Transient Suppressors in Automotive Applications

Special I/O interfacing is also required, because of the need for high tolerance to voltage transients, input noise, input/output isolation, etc. In addition, switches that are being monitored or driven by these buffers are usually referenced to chassis ground instead of signal ground, and in a car there can be many volts difference between the two. I/O interfacing is discussed in Reference 2.

Parting Thoughts

The main sources of information for this Application Note were the references by Ott and by White. Reference 5 is probably the finest treatment currently available on the subject. The other references provided specific information as cited in the text.

Courses and seminars on the subject of electromagnetic interference are given regularly throughout the year. Information on these can be obtained from:

IEEE Electromagnetic Compatibility Society
EMC Education Committee
345 East 47th Street
New York, NY 10017

Don White Consultants, Inc.
International Training Centre
P.O. Box D
Gainesville, VA 22065
Phone: (703) 347-0030

The EMC Education committee has available a video tape: "Introduction to EMC—A Video Training Tape," by Henry Ott. Don White Consultants offers a series of training courses on many different aspects of electromagnetic compatibility. Most organizations that sponsor EMC courses also offer in-plant presentations.

REFERENCES

1. Clark, O.M., "Electrostatic Discharge Protection Using Silicon Transient Suppressors," *Proceedings of the Electrical Overstress/Electrostatic Discharge Symposium.* Reliability Analysis Center, Rome Air Development Center, 1979.

2. Kearney, M; Shreve, J.; and Vincent, W., "Microprocessor Based Systems in the Automobile: Custom Integrated Circuits Provide an Effective Interface," *Electronic Engine Management and Driveline Control Systems,* SAE Publication SP-481, 810160, pp. 93–102.

3. King, W.M. and Reynolds, D., "Personnel Electrostatic Discharge: Impulse Waveforms Resulting From ESD of Humans Directly and Through Small Hand-Held Metallic Objects Intervening in the Discharge Path," *Proceedings of the IEEE Symposium on Electromagnetic Compatibility,* pp. 577-590, Aug. 1981.

4. Ott, H., "Digital Circuit Grounding and Interconnection," *Proceedings of the IEEE Symposium on Electromagnetic Compatibility,* pp. 292-297, Aug. 1981.

5. Ott, H., *Noise Reduction Techniques in Electronic Systems.* New York: Wiley, 1976.

6. *1981 Interference Technology Engineers' Master (ITEM) Directory and Design Guide.* R. and B. Enterprises, P.O. Box 328, Plymouth Meeting, PA 19426.

7. SAE J1211, "Recommended Environmental Practices for Electronic Equipment Design," *1980 SAE Handbook,* Part 1, pp. 22.80–22.96.

8. Smith, L., "A Watchdog Circuit for Microcomputer Based Systems," *Digital Design,* pp. 78, 79, Nov. 1979.

9. *TranZorb Quick Reference Guide.* General Semiconductor Industries, P.O. Box 3078, Tempe, AZ 85281.

10. Tucker, T.J., "Spark Initiation Requirements of a Secondary Explosive," *Annals of the New York Academy of Sciences,* Vol 152, Article I, pp. 643–653, 1968.

11. White, D., *Electromagnetic Interference and Compatibility, Vol. 3: EMI Control Methods and Techniques.* Don White Consultants, 1973.

12. White, D., *EMI Control in the Design of Printed Circuit Boards and Backplanes.* Don White Consultants, 1981.

13. Yarkoni, B. and Wharton, J., "Designing Reliable Software for Automotive Applications," *SAE Transactions,* 790237, July 1979.

APPLICATION NOTE

AP-155

Oscillators for Microcontrollers

TOM WILLIAMSON
MICROCONTROLLER
TECHNICAL MARKETING

December 1986

Order Number: 230659-001

INTRODUCTION

Intel's microcontroller families (MCS®-48, MCS®-51, and iACX-96) contain a circuit that is commonly referred to as the "on-chip oscillator". The on-chip circuitry is not itself an oscillator, of course, but an amplifier that is suitable for use as the amplifier part of a feedback oscillator. The data sheets and Microcontoller Handbook show how the on-chip amplifier and several off-chip components can be used to design a working oscillator. With proper selection of off-chip components, these oscillator circuits will perform better than almost any other type of clock oscillator, and by almost any criterion of excellence. The suggested circuits are simple, economical, stable, and reliable.

We offer assistance to our customers in selecting suitable off-chip components to work with the on-chip oscillator circuitry. It should be noted, however, that Intel cannot assume the responsibility of writing specifications for the off-chip components of the complete oscillator circuit, nor of guaranteeing the performance of the finished design in production, anymore than a transistor manufacturer, whose data sheets show a number of suggested amplifier circuits, can assume responsibility for the operation, in production, of any of them.

We are often asked why we don't publish a list of required crystal or ceramic resonator specifications, and recommend values for the other off-chip components. This has been done in the past, but sometimes with consequences that were not intended.

Suppose we suggest a maximum crystal resistance of 30 ohms for some given frequency. Then your crystal supplier tells you the 30-ohm crystals are going to cost twice as much as 50-ohm crystals. Fearing that Intel will not "guarantee operation" with 50-ohm crsytals, you order the expensive ones. In fact, Intel guarantees only what is embodied within an Intel product. Besides, there is no reason why 50-ohm crystals couldn't be used, if the other off-chip components are suitably adjusted.

Should we recommend values for the other off-chip components? Should we do it for 50-ohm crystals or 30-ohm crystals? With respect to what should we optimize their selection? Should we minimize start-up time or maximize frequency stability? In many applications, neither start-up time nor frequency stability are particularly critical, and our "recommendations" are only restricting your system to unnecessary tolerances. It all depends on the application.

Although we will neither "specify" nor "recommend" specific off-chip components, we do offer assistance in these tasks. Intel application engineers are available to provide whatever technical assistance may be needed or desired by our customers in designing with Intel products.

This Application Note is intended to provide such assistance in the design of oscillator circuits for microcontroller systems. Its purpose is to describe in a practical manner how oscillators work, how crystals and ceramic resonators work (and thus how to spec them), and what the on-chip amplifier looks like electronically and what its operating characteristics are. A BASIC program is provided in Appendix II to assist the designer in determining the effects of changing individual parameters. Suggestions are provided for establishing a pre-production test program.

FEEDBACK OSCILLATORS

Loop Gain

Figure 1 shows an amplifier whose output line goes into some passive network. If the input signal to the amplifier is v_1, then the output signal from the amplifer is $v_2 = Av_1$ and the output signal from the passive network is $v_3 = \beta v_2 = \beta A v_1$. Thus βA is the overall gain from terminal 1 to terminal 3.

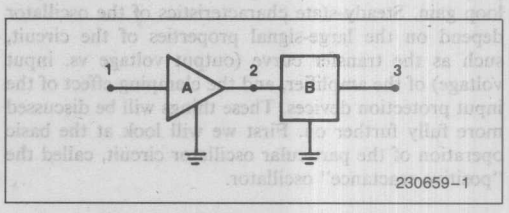


Figure 1. Factors in Loop Gain

Now connect terminal 1 to terminal 3, so that the signal path forms a loop: 1 to 2 to 3, which is also 1. Now we have a feedback loop, and the gain factor βA is called the *loop gain.*

Gain factors are complex numbers. That means they have a magnitude and a phase angle, both of which vary with frequency. When writing a complex number, one must specify both quantities, magnitude and angle. A number whose magnitude is 3, and whose angle is 45 degrees is commonly written this way: $3\angle 45°$. The number 1 is, in complex number notation, $1\angle 0°$, while -1 is $1\angle 180°$.

By closing the feedback loop in Figure 1, we force the equality

$$v_1 = \beta A v_1$$

This equation has two solutions:

1) $v_1 = 0$;

2) $\beta A = 1\angle 0°$.

In a given circuit, either or both of the solutions may be in effect. In the first solution the circuit is quiescent (no output signal). If you're trying to make an oscillator, a no-signal condition is unacceptable. There are ways to guarantee that the second solution is the one that will be in effect, and that the quiescent condition will be excluded.

How Feedback Oscillators Work

A feedback oscillator amplifies its own noise and feeds it back to itself in exactly the right phase, at the oscillation frequency, to build up and reinforce the desired oscillations. Its ability to do that depends on its loop gain. First, oscillations can occur only at the frequency for which the loop gain has a phase angle of 0 degrees. Second build-up of oscillations will occur only if the loop gain exceeds 1 at the frequency. Build-up continues until nonlinearities in the circuit reduce the average value of the loop gain to exactly 1.

Start-up characteristics depend on the small-signal properties of the circuit, specifically, the small-signal loop gain. Steady-state characteristics of the oscillator depend on the large-signal properties of the circuit, such as the transfer curve (output voltage vs. input voltage) of the amplifier, and the clamping effect of the input protection devices. These things will be discussed more fully further on. First we will look at the basic operation of the particular oscillator circuit, called the "positive reactance" oscillator.

The Positive Reactance Oscillator

Figure 2 shows the configuration of the positive reactance oscillator. The inverting amplifier, working into the impedance of the feedback network, produces an output signal that is nominally 180 degrees out of phase with its input. The feedback network must provide an additional 180 degrees phase shift, such that the overall loop gain has zero (or 360) degrees phase shift at the oscillation frequency.

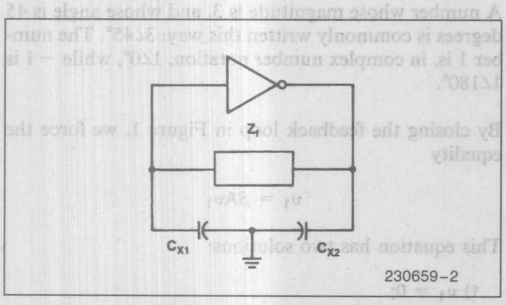


Figure 2. Positive Reactance Oscillator

In order for the loop gain to have zero phase angle it is necessary that the feedback element Z_f have a positive reactance. That is, it must be inductive. Then, the frequency at which the phase angle is zero is approximately the frequency at which

$$X_f = \frac{+1}{\omega C}$$

where X_f is the reactance of Z_f (the total Z_f being $R_f + jX_f$, and C is the series combination of C_{X1} and C_{X2}.

$$C = \frac{C_{X1}\,C_{X2}}{C_{X1} + C_{X2}}$$

In other words, Z_f and C form a parallel resonant circuit.

If Z_f is an inductor, then $X_f = \omega L$, and the frequency at which the loop gain has zero phase is the frequency at which

$$\omega L = \frac{1}{\omega C}$$

or

$$\omega = \frac{1}{\sqrt{LC}}$$

Normally, Z_f is not an inductor, but it must still have a positive reactance in order for the circuit to oscillate. There are some piezoelectric devices on the market that show a positive reactance, and provide a more stable oscillation frequency than an inductor will. Quartz crystals can be used where the oscillation frequency is critical, and lower cost ceramic resonators can be used where the frequency is less critical.

When the feedback element is a piezoelectric device, this circuit configuration is called a Pierce oscillator. The advantage of piezoelectric resonators lies in their property of providing a wide range of positive reactance values over a very narrow range of frequencies. The reactance will equal $1/\omega C$ at some frequency within this range, so the oscillation frequency will be within the same range. Typically, the width of this range is

only 0.3% of the nominal frequency of a quartz crystal, and about 3% of the nominal frequency of a ceramic resonator. With relatively little design effort, frequency accuracies of 0.03% or better can be obtained with quartz crystals, and 0.3% or better with ceramic resonators.

QUARTZ CRYSTALS

The crystal resonator is a thin slice of quartz sandwiched between two electrodes. Electrically, the device looks pretty much like a 5 or 6 pF capacitor, except that over certain ranges of frequencies the crystal has a positive (i.e., inductive) reactance.

The ranges of positive reactance originate in the piezoelectric property of quartz: Squeezing the crystal generates an internal E-field. The effect is reversible: Applying an *AC* E-field causes the crystal to vibrate. At certain vibrational frequencies there is a mechanical resonance. As the E-field frequency approaches a frequency of mechanical resonance, the measured reactance of the crystal becomes positive, as shown in Figure 3.

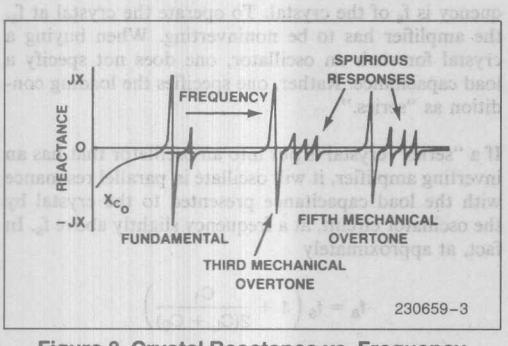


Figure 3. Crystal Reactance vs. Frequency

Typically there are several ranges of frequencies wherein the reactance of the crystal is positive. Each range corresponds to a different mode of vibration in the crystal. The main resonsances are the so-called fundamental response and the third and fifth overtone responses.

The overtone responses shouldn't be confused with the harmonics of the fundamental. They're not harmonics, but different vibrational modes. They're not in general at exact integer multiples of the fundamental frequency. There will also be "spurious" responses, occurring typically a few hundred KHz above each main response.

To assure that an oscillator starts in the desired mode on power-up, something must be done to suppress the loop gain in the undesired frequency ranges. The crystal itself provides some protection against unwanted modes of oscillation; too much resistance in that mode, for example. Additionally, junction capacitances in the amplifying devices tend to reduce the gain at higher frequencies, and thus may discriminate against unwanted modes. In some cases a circuit fix is necessary, such as inserting a trap, a phase shifter, or ferrite beads to kill oscillations in unwanted modes.

Crystal Parameters

Equivalent Circuit

Figure 4 shows an equivalent circuit that is used to represent the crystal for circuit analysis.

The R_1-L_1-C_1 branch is called the motivational arm of the crystal. The values of these parameters derive from the mechanical properties of the crystal and are constant for a given mode of vibration. Typical values for various nominal frequencies are shown in Table 1.

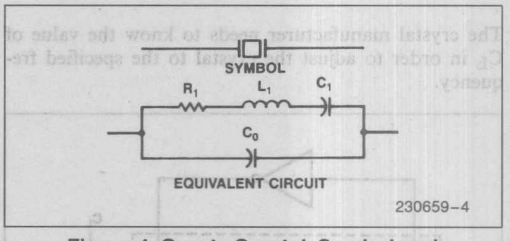


Figure 4. Quartz Crystal: Symbol and Equivalent Circuit

C_0 is called the shunt capacitance of the crystal. This is the capacitance of the crystal's electrodes and the mechanical holder. If one were to measure the reactance of the crystal at a freuqency far removed from a resonance frequency, it is the reactance of this capacitance that would be measured. It's normally 3 to 7 pF.

Table 1. Typical Crystal Parameters

Frequency MHz	R_1 ohms	L_1 mH	C_1 pF	C_0 pF
2	100	520	0.012	4
4.608	36	117	0.010	2.9
11.25	19	8.38	0.024	5.4

The series resonant frequency of the crystal is the frequency at which L_1 and C_1 are in resonance. This frequency is given by

$$f_s = \frac{1}{2\pi\sqrt{L_1 C_1}}$$

At this frequency the impedance of the crystal is R_1 in parallel with the reactance of C_0. For most purposes, this impedance is taken to be just R_1, since the reactance of C_0 is so much larger than R_1.

Load Capacitance

A crystal oscillator circuit such as the one shown in Figure 2 (redrawn in Figure 5) operates at the frequency for which the crystal is antiresonant (ie, parallel-resonant) with the total capacitance across the crystal terminals external to the crystal. This total capacitance external to the crystal is called the load capacitance.

As shown in Figure 5, the load capacitance is given by

$$C_L = \frac{C_{X1}\, C_{X2}}{C_{X1} + C_{X2}} + C_{stray}$$

The crystal manufacturer needs to know the value of C_L in order to adjust the crystal to the specified frequency.

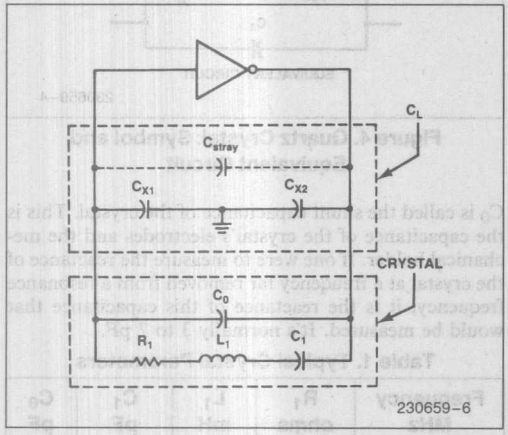


Figure 5. Load Capacitance

The adjustment involves putting the crystal in *series* with the specified C_L, and then "trimming" the crystal to obtain resonance of the series combination of the crystal and C_L at the specified frequency. Because of the high Q of the crystal, the resonant frequency of the series combination of the crystal and C_L is the same as the antiresonant frequency of the *parallel* combination of the crystal and C_L. This frequency is given by

$$f_a = \frac{1}{2\pi\sqrt{L_1 C_1 (C_L + C_0)/(C_1 + C_L + C_0)}}$$

These frequency formulas are derived (in Appendix A) from the equivalent circuit of the crystal, using the assumptions that the Q of the crystal is extremely high, and that the circuit external to the crystal has no effect on the frequency other than to provide the load capacitance C_L. The latter assumption is not precisely true, but it is close enough for present purposes.

"Series" vs. "Parallel" Crystals

There is no such thing as a "series cut" crystal as opposed to a "parallel cut" crystal. There are different cuts of crystal, having to do with the parameters of its motional arm in various frequency ranges, but there is no special cut for series or parallel operation.

An oscillator is series resonant if the oscillation frequency is f_s of the crystal. To operate the crystal at f_s, the amplifier has to be noninverting. When buying a crystal for such an oscillator, one does not specify a load capacitance. Rather, one specifies the loading condition as "series."

If a "series" crystal is put into an oscillator that has an inverting amplifier, it will oscillate in parallel resonance with the load capacitance presented to the crystal by the oscillator circuit, at a frequency slightly above f_s. In fact, at approximately

$$f_a = f_s\left(1 + \frac{C_1}{2(C_L + C_0)}\right)$$

This frequency would typically be about 0.02% above f_s.

Equivalent Series Resistance

The "series resistance" often listed on quartz crystal data sheets is the real part of the crystal impedance at the crystal's calibration frequency. This will be R1 if the calibration frequency is the series resonant frequency of the crystal. If the crystal is calibrated for parallel resonance with a load capacitance CL, the equivalent series resistance will be

$$ESR = R_1\left(1 + \frac{C_0}{C_L}\right)^2$$

The crystal manufacturer measures this resistance at the calibration frequency during the same operation in which the crystal is adjusted to the calibration frequency.

Frequency Tolerance

Frequency tolerance as discussed here is not a requirement on the crystal, but on the complete oscillator. There are two types of frequency tolerances on oscillators: frequency *acccuracy* and frequency *stability.* Frequency accuracy refers to the oscillator's ability to run at an exact specified frequency. Frequency stability refers to the constancy of the oscillation frequency.

Frequency accuracy requires mainly that the oscillator circuit present to the crystal the same load capacitance that it was adjusted for. Frequency stability requires mainly that the load capacitance be constant.

In most digital applications the accuracy and stability requirements on the oscillator are so wide that it makes very little difference what load capacitance the crystal was adjusted to, or what load capacitance the circuit actually presents to the crystal. For example, if a crystal was calibrated to a load capacitance of 25 pF, and is used in a circuit whose actual load capacitance is 50 pF, the frequency error on that account would be less than 0.01%.

In a positive reactance oscillator, the crystal only needs to be in the intended response mode for the oscillator to satisfy a 0.5% or better frequency tolerance. That's because for any load capacitance the oscillation frequency is certain to be between the crystal's resonant and antiresonant frequencies.

Phase shifts that take place within the amplifier part of the oscillator will also affect frequency accuracy and stability. These phase shifts can normally be modeled as an "output capacitance" that, in the positive reactance oscillator, parallels C_{X2}. The predictability and constancy of this output capacitance over temperature and device sample will be the limiting factor in determining the tolerances that the circuit is capable of holding.

Drive Level

Drive level refers to the power dissipation in the crystal. There are two reasons for specifying it. One is that the parameters in the equivalent circuit are somewhat dependent on the drive level at which the crystal is calibrated. The other is that if the application circuit exceeds the test drive level by too much, the crystal may be damaged. Note that the terms "test drive level" and "rated drive level" both refer to the drive level at which the crystal is calibrated. Normally, in a microcontroller system, neither the frequency tolerances nor the power levels justify much concern for this specification. Some crystal manufacturers don't even require it for microprocessor crystals.

In a positive reactance oscillator, if one assumes the peak voltage across the crystal to be something in the neighborhood of V_{CC}, the power dissipation can be approximated as

$$P = 2R_1 [\pi f (C_L + C_0) V_{CC}]^2$$

This formula is derived in Appendix A. In a 5V system, P rarely evaluates to more than a milliwatt. Crystals with a standard 1 or 2 mW drive level rating can be used in most digital systems.

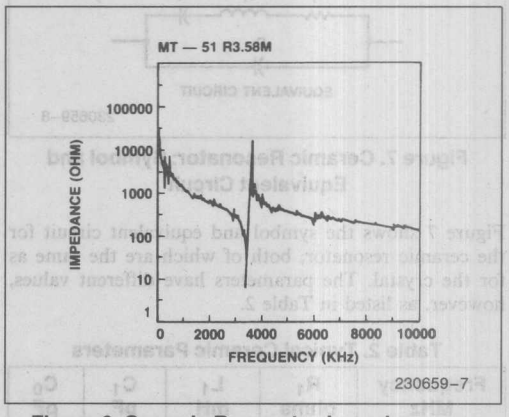


Figure 6. Ceramic Resonator Impedance vs. Frequency (Test Data Supplied by NTK Technical Ceramics)

CERAMIC RESONATORS

Ceramic resonators operate on the same basic principles as a quartz crsytal. Like quartz crsytals, they are piezoelectric, have a reactance versus frequency curve similar to a crystal's, and an equivalent circuit that looks just like a crystal's (with different parameter values, however).

The frequency tolerance of a ceramic resonator is about two orders of magnitude wider than a crystal's, but the ceramic is somewhat cheaper than a crystal. It may be noted for comparison that quartz crystals with relaxed tolerances cost about twice as much as ceramic resonators. For purposes of clocking a microcontroller, the frequency tolerance is often relatively noncritical, and the economic consideration becomes the dominant factor.

Figure 6 shows a graph of impedance magnitude versus frequency for a 3.58 MHz ceramic resonator. (Note that Figure 6 is a graph of $|Z_f|$ versus frequency, where

as Figure 3 is a graph of X_f versus frequency.) A number of spurious responses are apparent in Figure 6. The manufacturers state that spurious responses are more prevalent in the lower frequency resonators (kHz range) than in the higher frequency units (MHz range). For our purposes only the MHz range ceramics need to be considered.

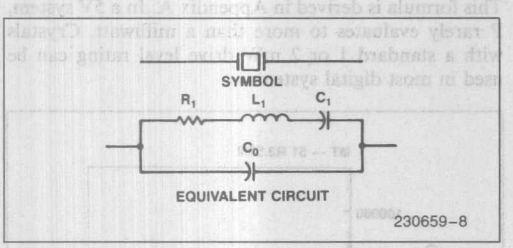


Figure 7. Ceramic Resonator: Symbol and Equivalent Circuit

Figure 7 shows the symbol and equivalent circuit for the ceramic resonator, both of which are the same as for the crystal. The parameters have different values, however, as listed in Table 2.

Table 2. Typical Ceramic Parameters

Frequency MHz	R_1 ohms	L_1 mH	C_1 pF	C_0 pF
3.58	7	0.113	19.6	140
6.0	8	0.094	8.3	60
8.0	7	0.092	4.6	40
11.0	10	0.057	3.9	30

Note that the motional arm of the ceramic resonator tends to have less resistance than the quartz crystal and also a vastly reduced L_1/C_1 ratio. This results in the motional arm having a Q (given by $(1/R_1)\sqrt{L_1/C_1}$) that is typically two orders of magnitude lower than that of a quartz crystal. The lower Q makes for a faster startup of the oscilaltor and for a less closely controlled frequency (meaning that circuitry external to the resonator will have more influence on the frequency than with a quartz crystal).

Another major difference is that the shunt capacitance of the ceramic resonator is an order of magnitude higher than C_0 of the quartz crystal and more dependent on the frequency of the resonator.

The implications of these differences are not all obvious, but some will be indicated in the section on Oscillator Calculations.

Specifications for Ceramic Resonators

Ceramic resonators are easier to specify than quartz crystals. All the vendor wants to know is the desired frequency and the chip you want it to work with. They'll supply the resonators, a circuit diagram showing the positions and values of other external components that may be required and a guarantee that the circuit will work properly at the specified frequency.

OSCILLATOR DESIGN CONSIDERATIONS

Designers of microcontroller systems have a number of options to choose from for clocking the system. The main decision is whether to use the "on-chip" oscillator or an external oscillator. If the choice is to use the on-chip oscillator, what kinds of external components are needed to make it operate as advertised? If the choice is to use an external oscillator, what type of oscillator should it be?

The decisions have to be based on both economic and technical requirements. In this section we'll discuss some of the factors that should be considered.

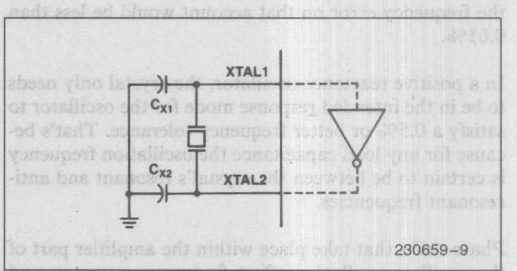


Figure 8. Using the "On-Chip" Oscillator

On-Chip Oscillators

In most cases, the on-chip amplifier with the appropriate external components provides the most economical solution to the clocking problem. Exceptions may arise in severe environments when frequency tolerances are tighter than about 0.01%.

The external components that need to be added are a positive reactance (normally a crystal or ceramic resonator) and the two capacitors C_{X1} and C_{X2}, as shown in Figure 8.

Crystal Specifications

Specifications for an appropriate crystal are not very critical, unless the frequency is. *Any* fundamental-mode crystal of medium or better quality can be used.

We are often asked what maximum crystal resistance should be specified. The best answer to this question is the lower the better, but use what's available. The crystal resistance will have some effect on start-up time and steady-state amplitude, but not so much that it can't be compensated for by appropriate selection of the capacitances C_{X1} and C_{X2}.

Similar questions are asked about specifications of load capacitance and shunt capacitance. The best advice we can give is to understand what these parameters mean and how they affect the operation of the circuit (that being the purpose of this Application Note), and then decide for yourself if such specifications are meaningful in your application or not. Normally, they're not, unless your frequency tolerances are tighter than about 0.1%.

Part of the problem is that crystal manufacturers are accustomed to talking "ppm" tolerances with radio engineers and simply won't take your order until you've filled out their list of specifications. It will help if you define your actual frequency tolerance requirements, both for yourself and to the crystal manufacturer. Don't pay for 0.003% crystals if your actual frequency tolerance is 1%.

Oscillation Frequency

The oscillation frequency is determined 99.5% by the crystal and up to about 0.5% by the circuit external to the crystal. The on-chip amplifier has little effect on the frequency, which is as it should be, since the amplifier parameters are temperature and process dependent.

The influence of the on-chip amplifier on the frequency is by means of its input and output (pin-to-ground) capacitances, which parallel C_{X1} and C_{X2}, and the XTAL1-to-XTAL2 (pin-to-pin) capacitance, which parallels the crystal. The input and pin-to-pin capacitances are about 7 pF each. Internal phase deviations from the nominal 180° can be modeled as an output capacitance of 25 to 30 pF. These deviations from the ideal have less effect in the positive reactance oscillator (with the inverting amplifer) than in a comparable series resonant oscillator (with the noninverting amplifier) for two reasons: first, the effect of the output capacitance is lessened, if not swamped, by the off-chip capacitor; secondly, the positive reactance oscillator is less sensitive, frequency-wise, to such phase errors.

Selection of C_{X1} and C_{X2}

Optimal values for the capacitors C_{X1} and C_{X2} depend on whether a quartz crystal or ceramic resonator is being used, and also on application-specific requirements on start-up time and frequency tolerance.

Start-up time is sometimes more critical in microcontroller systems than frequency stability, because of various reset and initialization requirements.

Less commonly, accuracy of the oscillator frequency is also critical, for example, when the oscillator is being used as a time base. As a general rule, fast start-up and stable frequency tend to pull the oscillator design in opposite directions.

Considerations of both start-up time and frequency stability over temperature suggest that C_{X1} and C_{X2} should be about equal and at least 20 pF. (But they don't *have* to be either.) Increasing the value of these capacitances above some 40 or 50 pF improves frequency stability. It also tends to increase the start-up time. There is a maximum value (several hundred pF, depending on the value of R_1 of the quartz or ceramic resonator) above which the oscillator won't start up at all.

If the on-chip amplifier is a simple inverter, such as in the 8051, the user can select values for C_{X1} and C_{X2} between some 20 and 100 pF, depending on whether start-up time or frequency stability is the more critical parameter in a specific application. If the on-chip amplifier is a Schmitt Trigger, such as in the 8048, smaller values of C_{X1} must be used (5 to 30 pF), in order to prevent the oscillator from running in a relaxation mode.

Later sections in this Application Note will discuss the effects of varying C_{X1} and C_{X2} (as well as other parameters), and will have more to say on their selection.

Placement of Components

Noise glitches arriving at XTAL1 or XTAL2 pins at the wrong time can cause a miscount in the internal clock-generating circuitry. These kinds of glitches can be produced through capacitive coupling between the oscillator components and PCB traces carrying digital signals with fast rise and fall times. For this reason, the oscillator components should be mounted close to the chip and have short, direct traces to the XTAL1, XTAL2, and VSS pins.

Clocking Other Chips

There are times when it would be desirable to use the on-chip oscillator to clock other chips in the system.

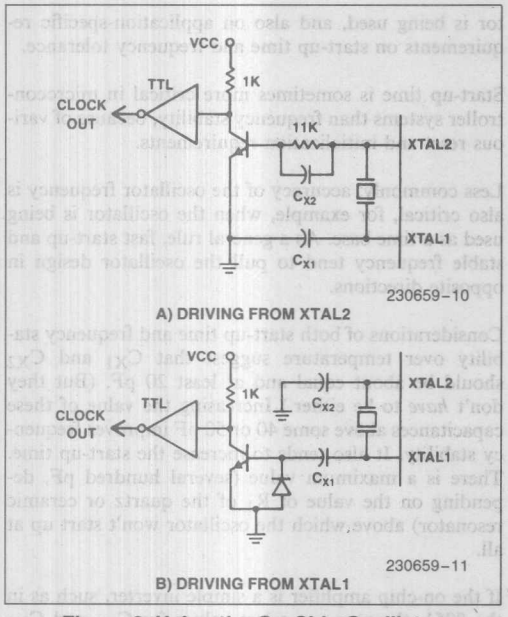


Figure 9. Using the On-Chip Oscillator to Drive Other Chips

This can be done if an appropriate buffer is used. A TTL buffer puts too much load on the on-chip amplifier for reliable start-up. A CMOS buffer (such as the 74HC04) can be used, if it's fast enough and if its VIH and VIL specs are compatible with the available signal amplitudes. Circuits such as shown in Figure 9 might also be considered for these types of applications.

Clock-related signals are available at the TO pin in the MCS-48 products, at ALE in the MCS-48 and MCS-51 lines, and the iACX-96 controllers provide a CLKOUT signal.

External Oscillators

When technical requirements dictate the use of an external oscillator, the external drive requirements for the microcontroller, as published in the data sheet, must be carefully noted. The logic levels are not in general TTL-compatible. And each controller has its idiosyncracies in this regard. The 8048, for example, requires that both XTAL1 and XTAL2 be driven. The 8051 *can* be driven that way, but the data sheet suggest the simpler method of grounding XTAL1 and driving XTAL2. For this method, the driving source must be capable of sinking some current when XTAL2 is being driven low.

For the external oscillator itself, there are basically two choices: ready-made and home-grown.

TTL Crystal Clock Oscillator

The HS-100, HS-200, & HS-500 all-metal package series of oscillators are TTL compatible & fit a DIP layout. Standard electrical specifications are shown below. Variations are available for special applications.

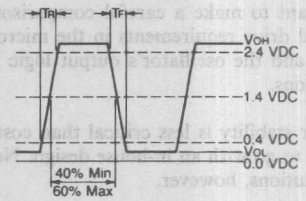
Frequency Range: HS-100—3.5 MHz to 30 MHz
HS-200—225 KHz to 3.5 MHz
HS-500—25 MHz to 60 MHz

Frequency Tolerance: ±0.1% Overall 0°C–70°C

Hermetically Sealed Package

Mass spectrometer leak rate max.
1×10^{-8} atmos. cc/sec. of helium

Output Waveform



230659-12

	HS-100		HS-200	HS-500
INPUT				
	3.5 MHz–20 MHz	20+ MHz–30 MHz	225 KHz–4.0 MHz	25 MHz–60 MHz
Supply Voltage (V_{CC})	5V ±10%	5V ±10%	5V ±10%	5V ±10%
Supply Current (I_{CC}) max.	30 mA	40 mA	85 mA	50 mA
OUTPUT				
	HS-100		**HS-200**	**HS-500**
	3.5 MHz–20 MHz	20+ MHz–30 MHz	225 KHz–4.0 MHz	25 MHz–60 MHz
V_{OH} (Logic "1")	+2.4V min.[1]	+2.7V min.[2]	+2.4V min.[1]	+2.7V min.[2]
V_{OL} (Logic "0")	+0.4V max.[3]	+0.5V max.[4]	+0.4V max.[3]	+0.5V max.[4]
Symmetry	60/40%[5]	60/40%[5]	55/45%[5]	60/40%[5]
T_R, T_F (Rise & Fall Time)	< 10 ns[6]	< 5 ns[6]	< 15 ns[6]	< 5 ns[6]
Output Short Circuit Current	18 mA min.	40 mA min.	18 mA min.	40 mA min.
Output Load	1 to 10 TTL Loads[7]	1 to 10 TTL Loads[8]	1 to 10 TTL Loads[7]	1 to 10 TTL Loads[8]

CONDITIONS

[1] I_O source = −400 μA max.
[2] I_O source = −1.0 mA max.
[3] I_O sink = 16.0 mA max.
[4] I_O sink = 20.00 mA max.
[5] V_O = 1.4V
[6] (0.4V to 2.4V)
[7] 1.6 mA per load
[8] 2.0 mA per load

Figure 10. Pre-Packaged Oscillator Data*

*Reprinted with the permission of ©Midland-Ross Corporation 1982.

Prepackaged oscillators are available from most crystal manufacturers, and have the advantage that the system designer can treat the oscillator as a black box whose performance is guaranteed by people who carry many years of experience in designing and building oscillators. Figure 10 shows a typical data sheet for some prepackaged oscillators. Oscillators are also available with complementary outputs.

If the oscillator is to drive the microcontroller directly, one will want to make a careful comparison between the external drive requirements in the microcontroller data sheet and the oscillator's output logic levels and test conditions.

If oscillator stability is less critical than cost, the user may prefer to go with an in-house design. Not without some precautions, however.

It's easy to design oscillators that work. Almost all of them do work, even if the designer isn't too clear on why. The key point here is that *almost* all of them work. The problems begin when the system goes into production, and marginal units commence malfunctioning in the field. Most digital designers, after all, are not very adept at designing oscillators *for production.*

Oscillator design is somewhat of a black art, with the quality of the finished product being *very* dependent on the designer's experience and intuition. For that reason the most important consideration in any design is to have an adequate preproduction test program. Preproduction tests are discussed later in this Application Note. Here we will discuss some of the design options and take a look at some commonly used configurations.

Gate Oscillators versus Discrete Devices

Digital systems designers are understandably reluctant to get involved with discrete devices and their peculiarities (biasing techniques, etc.). Besides, the component count for these circuits tends to be quite a bit higher than what a digital designer is used to seeing for that amount of functionality. Nevertheless, if there are unusual requirements on the accuracy and stability of the clock frequency, it should be noted that discrete device oscillators can be tailored to suit the exact needs of the application and perfected to a level that would be difficult for a gate oscillator to approach.

In most cases, when an external oscillator is needed, the designer tends to rely on some form of a gate oscillator. A TTL inverter with a resistor connecting the output to the input makes a suitable inverting amplifier. The resistor holds the inverter in the transition region between logical high and low, so that at least for start-up purposes the inverter is a linear amplifier.

The feedback resistance has to be quite low, however, since it must conduct current sourced by the input pin without allowing the DC input voltage to get too far above the DC output voltage. For biasing purposes, the feedback resistance should not exceed a few k-ohms. But shunting the crystal with such a low resistance does not encourage start-up.

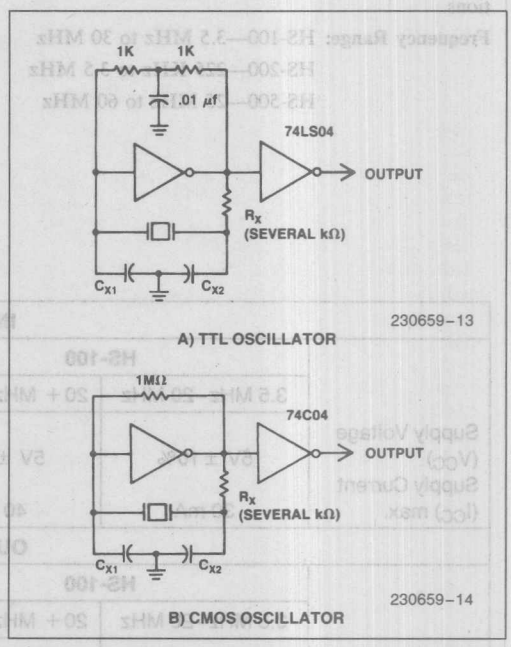


Figure 11. Commonly Used Gate Oscillators

Consequently, the configuration in Figure 11A might be suggested. By breaking R_f into two parts and AC-grounding the midpoint, one achieves the DC feedback required to hold the inverter in its active region, but without the negative signal feedback that is in effect telling the circuit *not* to oscillate. However, this biasing scheme will increase the start-up time, and relaxation-type oscillations are also possible.

A CMOS inverter, such as the 74HC04, might work better in this application, since a larger R_f can be used to hold the inverter in its linear region.

Logic gates tend to have a fairly low output resistance, which destabilizes the oscillator. For that reason a resistor Rx is often added to the feedback network, as shown in Figures 11A and B. At higher frequencies a 20 or 30 pF capacitor is sometimes used in the Rx position, to compensate for some of the internal propagation delay.

Reference 1 contains an excellent discussion of gate oscillators, and a number of design examples.

Fundamental versus Overtone Operation

It's easier to design an oscillator circuit to operate in the resonator's fundamental response mode than to design one for overtone operation. A quartz crystal whose fundamental response mode covers the desired frequency can be obtained up to some 30 MHz. For frequencies above that, the crystal might be used in an overtone mode.

Several problems arise in the design of an overtone oscillator. One is to stop the circuit from oscillating in the fundamental mode, which is what it would really rather do, for a number of reasons, involving both the amplifying device and the crystal. An additional problem with overtone operation is an increased tendency to spurious oscillations. That is because the R_1 of various spurious modes is likely to be about the same as R_1 of the intended overtone response. It may be necessary, as suggested in Reference 1, to specify a "spurious-to-main-response" resistance ratio to avoid the possibility of trouble.

Overtone oscillators are not to be taken lightly. One would be well advised to consult with an engineer who is knowledgeable in the subject during the design phase of such a circuit.

Series versus Parallel Operation

Series resonant oscillators use noninverting amplifiers. To make a noninverting amplifier out of logic gates requires that two inverters be used, as shown in Figure 12.

This type of circuit tends to be inaccurate and unstable in frequency over variations in temperature and V_{CC}. It has a tendency to oscillate at overtones, and to oscillate through C_0 of the crystal or some stray capacitance rather than as controlled by the mechanical resonance of the crystal.

The demon in series resonant oscillators is the phase shift in the amplifier. The series resonant oscillator wants more than just a "noninverting" amplifier—it wants a *zero phase-shift* amplifier. Multistage noninverting amplifiers tend to have a considerably lagging phase shift, such that the crystal reactance must be capacitive in order to bring the total phase shift around the feedback loop back up to 0. In this mode, a "12 MHz" crystal may be running at 8 or 9 MHz. One can put a capacitor in series with the crystal to relieve the crystal of having to produce all of the required phase shift, and bring the oscillation frequency closer to fs. However, to further complicate the situation, the amplifier's phase shift is strongly dependent on frequency, temperature, VCC, and device sample.

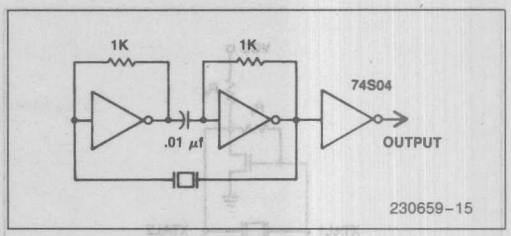


Figure 12. "Series Resonant" Gate Oscillator

Positive reactance oscillators ("parallel resonant") use inverting amplifiers. A single logic inverter can be used for the amplifier, as in Figure 11. The amplifier's phase shift is less critical, compared to a series resonant circuit, and since only one inverter is involved there's less phase error anyway. The oscillation frequency is effectively bounded by the resonant and antiresonant frequencies of the crystal itself. In addition, the feedback network includes capacitors that parallel the input and output terminals of the amplifier, thus reducing the effect of unpredictable capacitances at these points.

MORE ABOUT USING THE "ON-CHIP" OSCILLATORS

In this section we will describe the on-chip inverters on selected microcontrollers in some detail, and discuss criteria for selecting components to work with them. Future data sheets will supplement this discussion with updates and information pertinent to the use of each chip's oscillator circuitry.

Oscillator Calculations

Oscillator design, though aided by theory, is still largely an empirical exercise. The circuit is inherently nonlinear, and the normal analysis parameters vary with instantaneous voltage. In addition, when dealing with the on-chip circuitry, we have FETs being used as resistors, resistors being used as interconnects, distributed delays, input protection devices, parasitic junctions, and processing variations.

Consequently, oscillator calculations are never very precise. They can be useful, however, if they will at least indicate the effects of *variations* in the circuit parameters on start-up time, oscillation frequency, and steady-state amplitude. Start-up time, for example, can be taken as an indication of start-up reliability. If preproduction tests indicate a possible start-up problem, a relatively inexperienced designer can at least be made aware of what parameter may be causing the marginality, and what direction to go in to fix it.

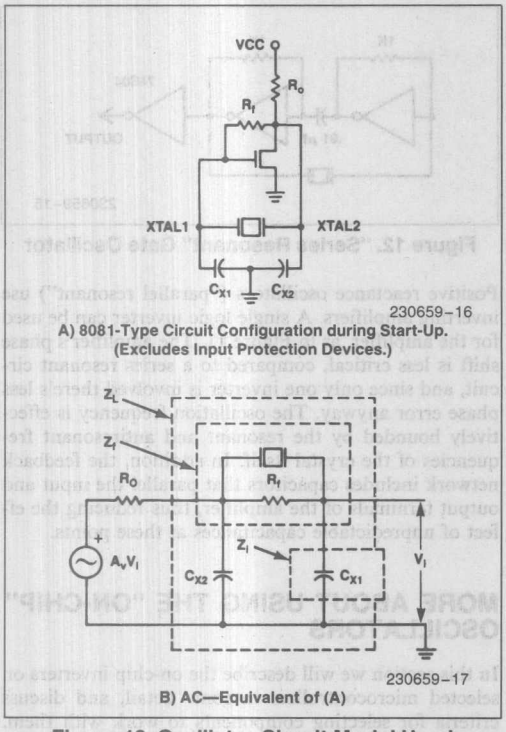


Figure 13. Oscillator Circuit Model Used in Start-Up Calculations

The analysis used here is mathematically straightforward but algebraically intractable. That means it's relatively easy to understand and program into a computer, but it will not yield a neat formula that gives, say, steady-state amplitude as a function of this or that list of parameters. A listing of a BASIC program that implements the analysis will be found in Appendix II.

When the circuit is first powered up, and before the oscillations have commenced (and if the oscillations *fail* to commence), the oscillator can be treated as a small signal linear amplifier with feedback. In that case, standard small-signal analysis techniques can be used to determine start-up characteristics. The circuit model used in this analysis is shown in Figure 13.

The circuit approximates that there are no high-frequency effects within the amplifier itslef, such that its high-frequency behavior is dominated by the load impedance Z_L. This is a reasonable approximation for single-stage amplifiers of the type used in 8051-type devices. Then the gain of the amplifier as a function of frequency is

$$A = \frac{A_v Z_L}{Z_L + R_0}$$

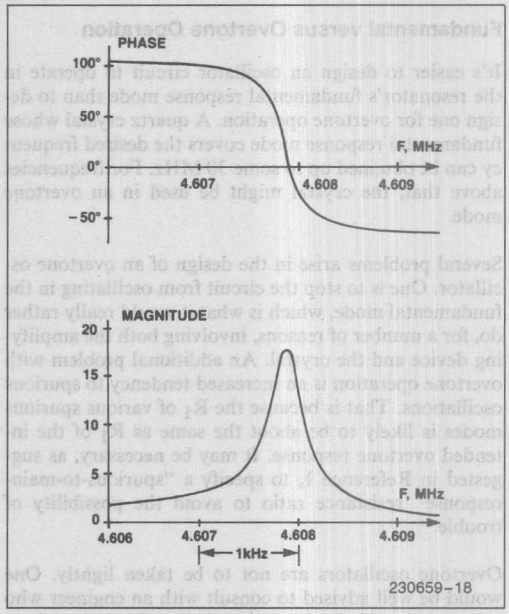


Figure 14. Loop Gain versus Frequency (4.608 MHz Crystal)

The gain of the feedback network is

$$\beta = \frac{Z_i}{Z_i + Z_f}$$

And the loop gain is

$$\beta A = \frac{Z_i}{Z_i + Z_f} \times \frac{A_v Z_L}{Z_L + R_0}$$

The impedances Z_L, Z_f, and Z_i are defined in Figure 13B.

Figure 14 shows the way the loop gain thus calculated (using typical 8051-type parameters and a 4.608 MHz crystal) varies with frequency. The frequency of interest is the one for which the phase of the loop gain is zero. The accepted criterion for start-up is that the magnitude of the loop gain must exceed unity at this frequency. This is the frequency at which the circuit is in resonance. It corresponds very closely with the antiresonant frequency of the motional arm of the crystal in parallel with C_L.

Figure 15 shows the way the loop gain varies with frequency when the parameters of a 3.58 MHz ceramic resonator are used in place of a crystal (the amplifier parameters being typical 8051, as in Figure 14). Note the different frequency scales.

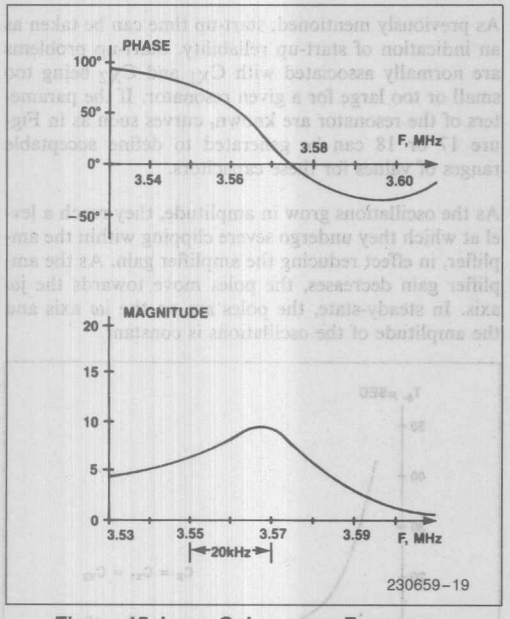


Figure 15. Loop Gain versus Frequency (3.58 MHz Ceramic)

Start-Up Characteristics

It is common, in studies of feedback systems, to examine the behavior of the closed loop gain as a function of complex frequency $s = \sigma + j\omega$; specifically, to determine the location of its poles in the complex plane. A pole is a point on the complex plane where the gain function goes to infinity. Knowledge of its location can be used to predict the response of the system to an input disturbance.

The way that the response function depends on the location of the poles is shown in Figure 16. Poles in the left-half plane cause the response function to take the form of a damped sinusoid. Poles in the right-half plane cause the response function to take the form of an exponentially growing sinusoid. In general,

$$v(t) \sim e^{at} \sin(\omega t + \theta)$$

where *a* is the real part of the pole frequency. Thus if the pole is in the right-half plane, a is positive and the sinusoid grows. If the pole is in the left-half plane, a is negative and the sinusoid is damped.

The same type of analysis can usefully be applied to oscillators. In this case, however, rather than trying to ensure that the poles are in the left-half plane, we would seek to ensure that they're in the *right*-half plane. An exponentially growing sinusoid is exactly what is wanted from an oscillator that has just been powered up.

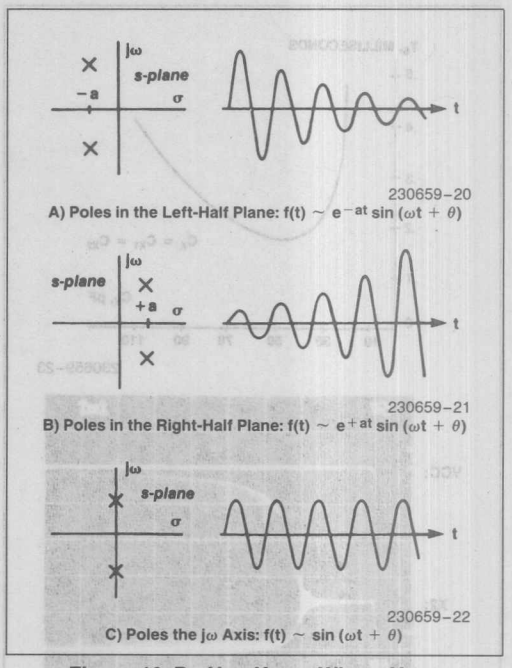


Figure 16. Do You Know Where Your Poles Are Tonight?

The gain function of interest in oscillators is $1/(1 - \beta A)$. Its poles are at the complex frequencies where $\beta A = 1\angle 0°$, because that value of βA causes the gain function to go to infinity. The oscillator will start up if the real part of the pole frequency is positive. More importantly, the *rate* at which it starts up is indicated by how *much* greater than 0 the real part of the pole frequency is.

The circuit in Figure 13B can be used to find the pole frequencies of the oscillator gain function. All that needs to be done is evaluate the impedances at complex frequencies $\sigma + j\omega$ rather than just at ω, and find the value of $\sigma + j\omega$ for which $\beta A = 1\angle 0°$. The larger that value of σ is, the faster the oscillator will start up.

Of course, other things besides pole frequencies, things like the VCC rise time, are at work in determining the start-up time. But to the extend that the pole frequencies *do* affect start-up time, we can obtain results like those in Figures 17 and 18.

To obtain these figures the pole frequencies were computed for various values of capacitance C_X from XTAL1 and XTAL2 to ground (thus $C_{X1} = C_{X2} = C_X$). Then a "time constant" for start-up was calculated as $T_s = \frac{1}{\sigma}$ where σ is the real part of the pole frequency (rad/sec), and this time constant is plotted versus C_X.

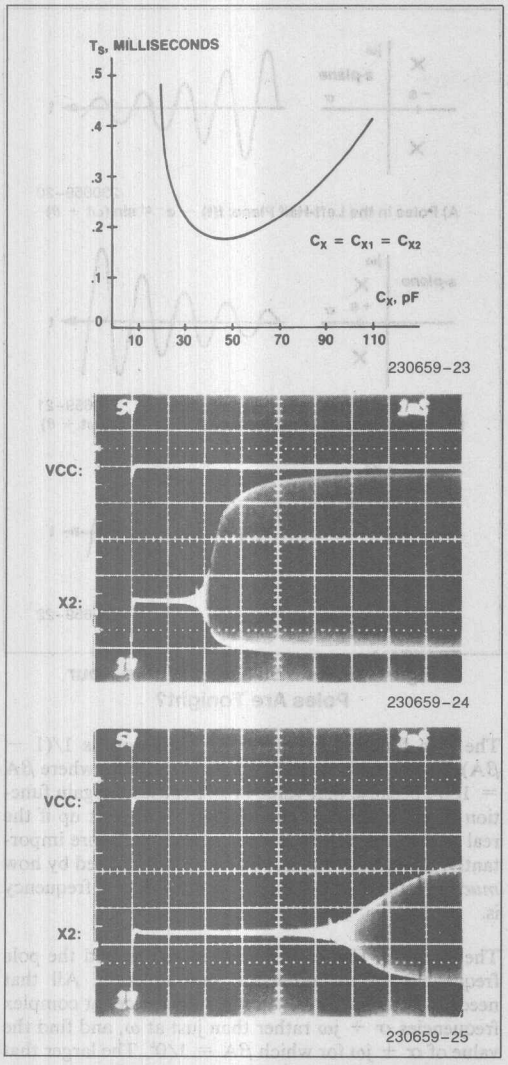


Figure 17. Oscillator Start-Up (4.608 MHz Crystal from Standard Crystal Corp.)

A short time constant means faster start-up. A long time constant means slow start-up. Observations of actual start-ups are shown in the figures. Figure 17 is for a typical 8051 with a 4.608 MHz crystal supplied by Standard Crystal Corp., and Figure 18 is for a typical 8051 with a 3.58 MHz ceramic resonator supplied by NTK Technical Ceramics, Ltd.

It can be seen in Figure 17 that, for this crystal, values of C_X between 30 and 50 pF minimize start-up time, but that the exact value in this range is not particularly important, even if the start-up time itself is critical.

As previously mentioned, start-up time can be taken as an indication of start-up reliability. Start-up problems are normally associated with C_{X1} and C_{X2} being too small or too large for a given resonator. If the parameters of the resonator are known, curves such as in Figure 17 or 18 can be generated to define acceptable ranges of values for these capacitors.

As the oscillations grow in amplitude, they reach a level at which they undergo severe clipping within the amplifier, in effect reducing the amplifier gain. As the amplifier gain decreases, the poles move towards the $j\omega$ axis. In steady-state, the poles are on the $j\omega$ axis and the amplitude of the oscillations is constant.

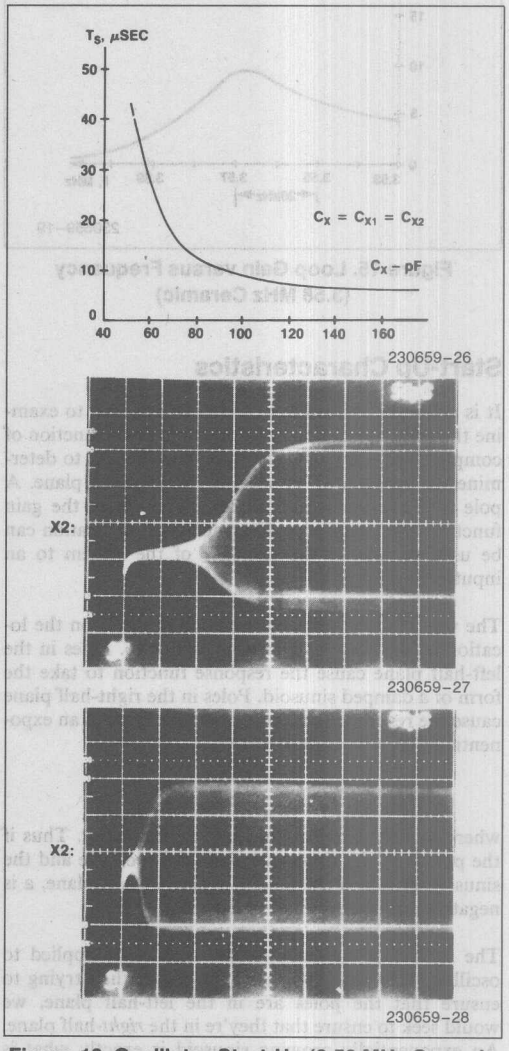


Figure 18. Oscillator Start-Up (3.58 MHz Ceramic Resonator from NTK Technical Ceramics)

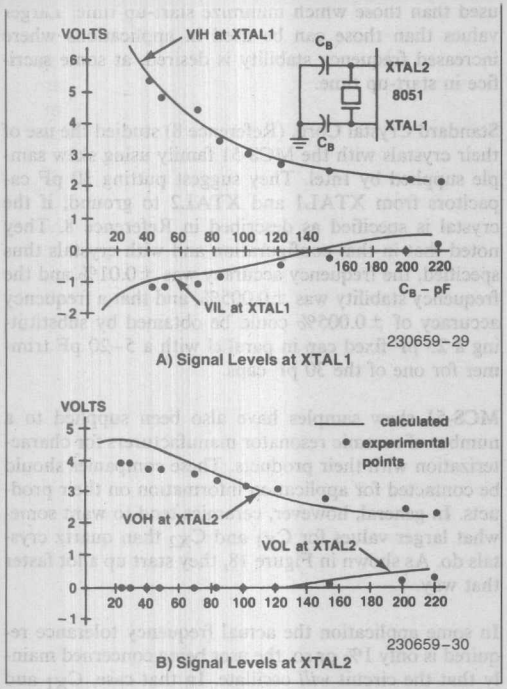


Figure 19. Calculated and Experimental Steady-State Amplitudes vs. Bulk Capacitance from XTAL1 and XTAL2 to Ground

Steady-State Characteristics

Steady-state analysis is greatly complicated by the fact that we are dealing with large signals and nonlinear circuit response. The circuit parameters vary with instantaneous voltage, and a number of clamping and clipping mechanisms come into play. Analyses that take all these things into account are too complicated to be of general use, and analyses that don't take them into account are too inaccurate to justify the effort.

There is a steady-state analysis in Appendix B that takes some of the complications into account and ignores others. Figure 19 shows the way the steady-state amplitudes thus calculated (using typical 8051 parameters and a 4.608 MHz crystal) vary with equal bulk capacitance placed from XTAL1 and XTAL2 to ground. Experimental results are shown for comparison.

The waveform at XTAL1 is a fairly clean sinusoid. Its negative peak is normally somewhat below zero, at a level which is determined mainly by the input protection circuitry at XTAL1.

The input protection circuitry consists of an ohmic resistor and an enhancement-mode FET with the gate and source connected to ground (VSS), as shown in Figure 20 for the 8051, and in Figure 21 for the 8048. Its function is to limit the positive voltage at the gate of the input FET to the avalanche voltage of the drain junction. If the input pin is driven below VSS, the drain and source of the protection FET interchange roles, so its gate is connected to what is now the drain. In this condition the device resembles a diode with the anode connected to VSS.

There is a parasitic pn junction between the ohmic resistor and the substrate. In the ROM parts (8015, 8048, etc.) the substrate is held at approximately $-3V$ by the on-chip back-bias generator. In the EPROM parts (8751, 8748, etc.) the substrate is connected to VSS.

The effect of the input protection circuitry on the oscillator is that if the XTAL1 signal goes negative, its negative peak is clamped to $-V_{DS}$ of the protection FET in the ROM parts, and to about $-0.5V$ in the EPROM parts. These negative voltages on XTAL1 are in this application self-limiting and nondestructive.

The clamping action does, however, raise the DC level at XTAL1, which in turn tends to reduce the positive peak at XTAL2. The waveform at XTAL2 resembles a sinusoid riding on a DC level, and whose negative peaks are clipped off at zero.

Since it's normally the XTAL2 signal that drives the internal clocking circuitry, the question naturally arises as to how large this signal must be to reliably do its job. In fact, the XTAL2 signal doesn't have to meet the same VIH and VIL specifications that an external driver would have to. That's because as long as the oscillator is working, the on-chip amplifier is driving itself through its own 0-to-1 transition region, which is very nearly the same as the 0-to-1 transition region in the internal buffer that follows the oscillator. If some processing variations move the transition level higher or lower, the on-chip amplifier tends to compensate for it by the fact that its own transition level is correspondingly higher or lower. (In the 8096, it's the XTAL1 signal that drives the internal clocking circuitry, but the same concept applies.)

The main concern about the XTAL2 signal amplitude is an indication of the general health of the oscillator. An amplitude of less than about 2.5V peak-to-peak indicates that start-up problems could develop in some units (with low gain) with some crystals (with high R_1). The remedy is to either adjust the values of C_{X1} and/or C_{X2} or use a crystal with a lower R_1.

The amplitudes at XTAL1 and XTAL2 can be adjusted by changing the ratio of the capacitors from XTAL1 and XTAL2 to ground. Increasing the XTAL2 capacitance, for example, decreases the amplitude at XTAL2 and increases the amplitude at XTAL1 by about the same amount. Decreasing both caps increases both amplitudes.

Pin Capacitance

Internal pin-to-ground and pin-to-pin capacitances at XTAL1 and XTAL2 will have some effect on the oscillator. These capacitances are normally taken to be in the range of 5 to 10 pF, but they are extremely difficult to evaluate. Any measurement of one such capacitance will necessarily include effects from the others. One advantage of the positive reactance oscillator is that the pin-to-ground capacitances are paralleled by external bulk capacitors, so a precise determination of their value is unnecessary. We would suggest that there is little justification for more precision than to assign them a value of 7 pF (XTAL1-to-ground and XTAL1-to-XTAL2). This value is probably not in error by more than 3 or 4 pF.

The XTAL2-to-ground capacitance is not entirely "pin capacitance," but more like an "equivalent output capacitance" of some 25 to 30 pF, having to include the effect of internal phase delays. This value will vary to some extent with temperature, processing, and frequency.

MCS®-51 Oscillator

The on-chip amplifier on the HMOS MCS-51 family is shown in Figure 20. The drain load and feedback "resistors" are seen to be field-effect transistors. The drain load FET, R_D, is typically equivalent to about 1K to 3 K-ohms. As an amplifier, the low frequency voltage gain is normally between -10 and -20, and the output resistance is effectively R_D.

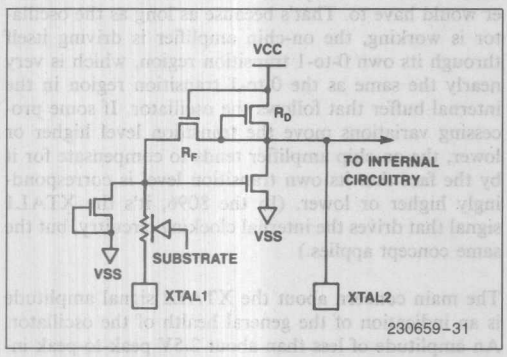


Figure 20. MCS®-51 Oscillator Amplifier

The 80151 oscillator is normally used with equal bulk capacitors placed externally from XTAL1 to ground and from XTAL2 to ground. To determine a reasonable value of capacitance to use in these positions, given a crystal of ceramic resonator of known parameters, one can use the BASIC analysis in Appendix II to generate curves such as in Figures 17 and 18. This procedure will define a range of values that will minimize start-up time. We don't suggest that smaller values be used than those which minimize start-up time. Larger values than those can be used in applications where increased frequency stability is desired, at some sacrifice in start-up time.

Standard Crystal Corp. (Reference 8) studied the use of their crystals with the MCS-51 family using skew sample supplied by Intel. They suggest putting 30 pF capacitors from XTAL1 and XTAL2 to ground, if the crystal is specified as described in Reference 8. They noted that in that configuration and with crystals thus specified, the frequency accuracy was $\pm 0.01\%$ and the frequency stability was $\pm 0.005\%$, and that a frequency accuracy of $\pm 0.005\%$ could be obtained by substituting a 25 pF fixed cap in parallel with a 5–20 pF trimmer for one of the 30 pF caps.

MCS-51 skew samples have also been supplied to a number of ceramic resonator manufacturers for characterization with their products. These companies should be contacted for application information on their products. In general, however, ceramics tend to want somewhat larger values for C_{X1} and C_{X2} than quartz crystals do. As shown in Figure 18, they start up a lot faster that way.

In some application the actual frequency tolerance required is only 1% or so, the user being concerned mainly that the circuit *will* oscillate. In that case, C_{X1} and C_{X2} can be selected rather freely in the range of 20 to 80 pF.

As you can see, "best" values for these components and their tolerances are strongly dependent on the application and its requirements. In any case, their suitability should be verified by environmental testing before the design is submitted to production.

MCS®-48 Oscillator

The NMOS and HMOS MCS-48 oscillator is shown in Figure 21. It differs from the 8051 in that its inverting

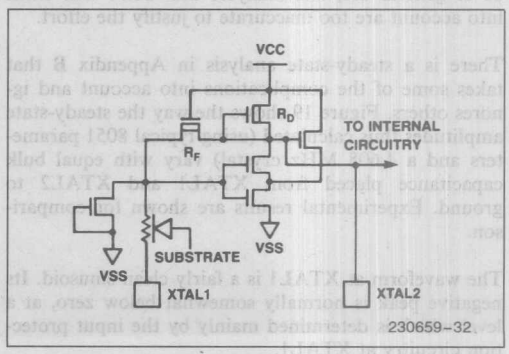


Figure 21. MCS®-48 Oscillator Amplifier

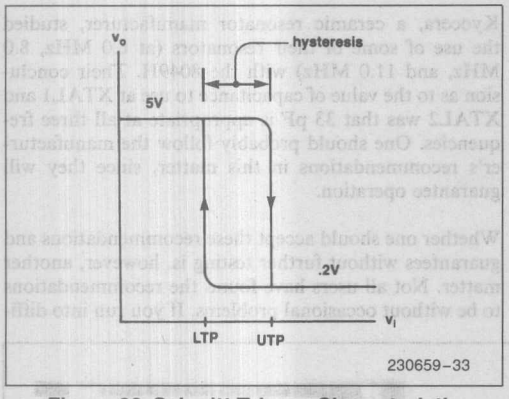


Figure 22. Schmitt Trigger Characteristic

amplifier is a Schmitt Trigger. This configuration was chosen to prevent crosstalk from the TO pin, which is adjacent to the XTAL1 pin.

All Schmitt Trigger circuits exhibit a hysteresis effect, as shown in Figure 22. The hysteresis is what makes it less sensitive to noise. The same hysteresis allows any Schmitt Trigger to be used as a relaxation oscillator. All you have to do is connect a resistor from output to input, and a capacitor from input to ground, and the circuit oscillates in a relaxation mode as follows.

If the Schmitt Trigger output is at a logic high, the capacitor commences charging through the feedback resistor. When the capacitor voltage reaches the upper trigger point (UTP), the Schmitt Trigger output switches to a logic low and the capacitor commences discharging through the same resistor. When the capacitor voltage reaches the lower trigger point (LTP), the Schmitt Trigger output switches to a logic high again, and the sequence repeats. The oscillation frequency is determined by the RC time constant and the hysteresis voltage, UTP-LTP.

The 8048 can oscillate in this mode. It has an internal feedback resistor. All that's needed is an external capacitor from XTAL1 to ground. In fact, if a smaller external feedback resistor is added, an 8048 system could be designed to run in this mode. *Do it at your own risk!* This mode of operation is not tested, specified, documented, or encouraged in any way by Intel for the 8048. Future steppings of the device might have a different type of inverting amplifier (one more like the 8051). The CHMOS members of the MCS-48 family do not use a Schmitt Trigger as the inverting amplifier.

Relaxation oscillations in the 8048 must be avoided, and this is the major objective in selecting the off-chip components needed to complete the oscillator circuit.

When an 8048 is powered up, if VCC has a short rise time, the relaxation mode starts first. The frequency is normally about 50 KHz. The resonator mode builds more slowly, but it eventually takes over and dominates the operation of the cirucit. This is shown in Figure 23A.

Due to processing variations, some units seem to have a harder time coming out of the relaxation mode, particularly at low temperatures. In some cases the resonator oscillations may fail entirely, and leave the device in the relaxation mode. Most units will stick in the relaxation mode at any temperature if C_{X1} is larger than about 50 pF. Therefore, C_{X1} should be chosen with some care, particularly if the system must operate at lower temperatures.

One method that has proven effective in all units to −40°C is to put 5 pF from XTAL1 to ground and 20 pF from XTAL2 to ground. Unfortunately, while this method does discourage the relaxation mode, it is not an optimal choice for the resonator mode. For one thing, it does not swamp the pin capacitance. Also, it makes for a rather high signal level at XTAL1 (8 or 9 volts peak-to-peak).

The question arises as to whether that level of signal at XTLA1 might damage the chip. Not to worry. The negative peaks are self-limiting and nondestructive. The positive peaks could conceivably damage the oxide, but in fact, NMOS chips (eg, 8048) and HMOS chips (eg, 8048H) are tested to a much higher voltage than that. The technology trend, of course, is to thinner oxides, as the devices shrink in size. For an extra margin of safety, the HMOS II chips (eg, 8048AH) have an internal diode clamp at XTAL1 to VCC.

In reality, C_{X1} doesn't have to be quite so small to avoid relaxation oscillations, if the minimum operating temperature is not −40°C. For less severe temperature requirements, values of capacitance selected in much the same way as for an 8051 can be used. The circuit should be tested, however, at the system's lowest temperature limit.

Additional security against relaxation oscillations can be obtained by putting a 1M-ohm (or larger) resistor from XTAL1 to VCC. Pulling up the XTAL1 pin this way seems to discourage relaxation oscillations as effectively as any other method (Figure 23B).

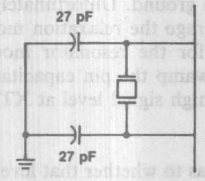
Another thing that discourages relaxation oscillations is low VCC. The resonator mode, on the other hand is much less sensitive to VCC. Thus if VCC comes up relatively slowly (several milliseconds rise time), the resonator mode is normally up and running before the relaxation mode starts (in fact, before VCC has even reached operating specs). This is shown in Figure 23C.

A secondary effect of the hysteresis is a shift in the oscillation frequency. At low frequencies, the output signal from an inverter without hysteresis leads (or lags) the input by 180 degrees. The hysteresis in a Schmitt Trigger, however, causes the output to lead the

input by less than 180 degrees (or lag by more than 180 degrees), by an amount that depends on the signal amplitude, as shown in Figure 24. At higher frequencies, there are additional phase shifts due to the various reactances in the circuit, but the phase shift due to the hysteresis is still present. Since the total phase shift in the oscillator's loop gain is necessarily 0 or 360 degrees, it is apparent that as the oscillations build up, the frequency has to change to allow the reactances to compensate for the hysteresis. In normal operation, this additional phase shift due to hysteresis does not exceed a few degrees, and the resulting frequency shift is negligible.

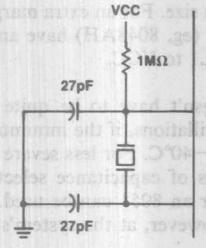
Kyocera, a ceramic resonator manufacturer, studied the use of some of their resonators (at 6.0 MHz, 8.0 MHz, and 11.0 MHz) with the 8049H. Their conclusion as to the value of capacitance to use at XTAL1 and XTAL2 was that 33 pF is appropriate at all three frequencies. One should probably follow the manufacturer's recommendations in this matter, since they will guarantee operation.

Whether one should accept these recommendations and guarantees without further testing is, however, another matter. Not all users have found the recommendations to be without occasional problems. If you run into diffi-



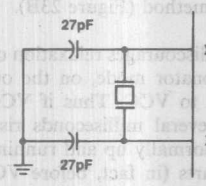
230659-34

A) When VCC Comes Up Fast, Relaxation Oscillations Start First. But Then the Crystal Takes Over.



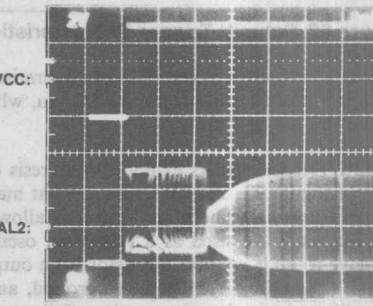
230659-35

B) Weak Pullup (1 MΩ or More) on XTAL1 Discourages Relaxation Mode.

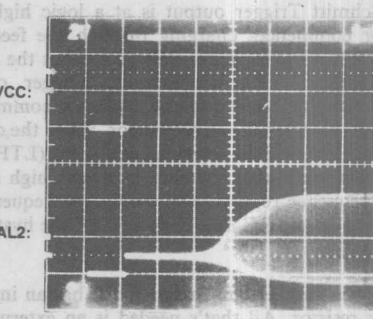


230659-36

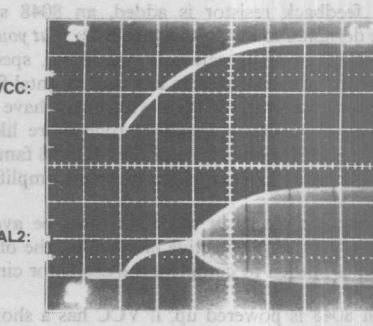
C) No Relaxation Oscillations When VCC Comes Up More Slowly.



230659-37



230659-38



230659-39

Figure 23. Relaxation Oscillations in the 8048

culties using their recommendations, both Intel and the ceramic resonator manufacturer want to know about it. It is to their interest, and ours, that such problems be resolved.

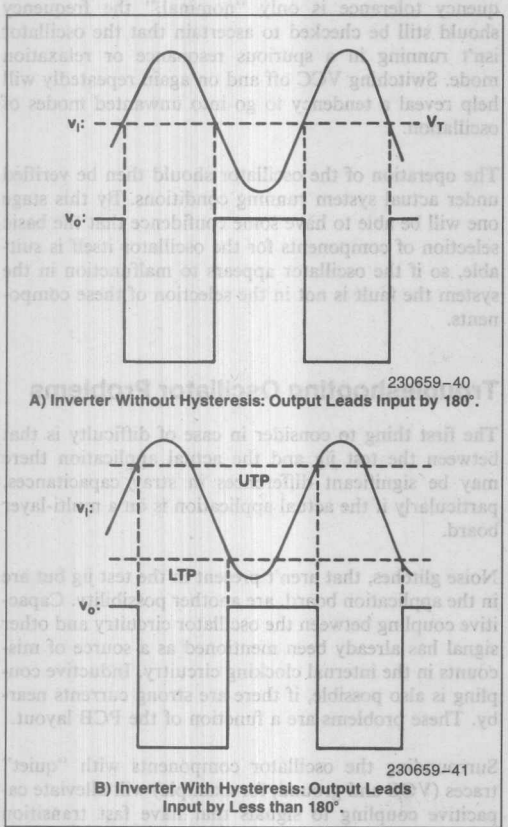


Figure 24. Amplitude—Dependent Phase Shift in Schmitt Trigger

Preproduction Tests

An oscillator design should never be considered ready for production until it has proven its ability to function acceptably well under worst-case environmental conditions and with parameters at their worst-case tolerance limits. Unexpected temperature effects in parts that may already be near their tolerance limits can prevent start-up of an oscillator that works perfectly well on the bench. For example, designers often overlook temperature effects in ceramic capacitors. (Some ceramics are down to 50% of their room-temperature values at −20°C and +60°C). The problem here isn't just one of frequency stability, but also involves start-up time and steady-state amplitude. There may also be temperature effects in the resonator and amplifier.

It will be helpful to build a test jig that will allow the oscillator circuit to be tested independently of the rest of the system. Both start-up and steady-state characteristics should be tested. Figure 25 shows the circuit that

A) Software for Oscillator Test

```
SOURCE
ORG 0000 H
        JMP   START
ORG 000B H                ;TIMER 0 INTERRUPT:
        CPL   T1          ;  TOGGLE T1
        RETI
ORG 001BH                 ;TIMER 1 INTERRUPT:
        CPL   P1.1        ;  TOGGLE CRO TRIGGER
        DJNZ  P2,$        ;  DELAY
        CPL   P1.0        ;  TOGGLE VCC CONTROL
        RETI
START:  MOV   TH1,#0FAH ;TIMER 1 RELOAD VALUE
        MOV   TL1,#0FAH ;START TL1 AT RELOAD VALUE
        MOV   TMOD,#61H ;TIMER 1 TO COUNTER, AUTO
                        ;RELOAD
                        ;TIMER 0 TO TIMER, 16-BIT
        MOV   IE,#BAH   ;ENABLE TIMER INTERRUPTS
                        ;ONLY
        MOV   TCON,#50H ;TURN ON BOTH TIMERS
        JMP   $         ;IDLE
END
```

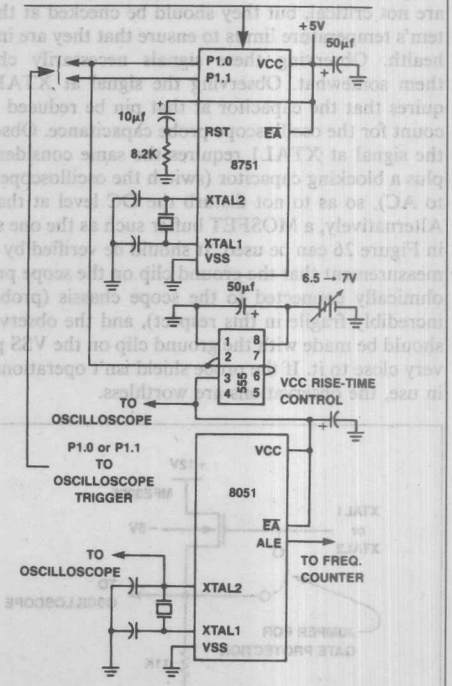


Figure 25. Oscillator Test Circuit and Software

was used to obtain the oscillator start-up photographs in this Application Note. This circuit or a modified version of it would make a convenient test vehicle. The oscillator and its relevant components can be physically separated from the control circuitry, and placed in a temperature chamber.

Start-up should be observed under a variety of conditions, including low VCC and using slow and fast VCC rise times. The oscillator should not be reluctant to start up even when VCC is below its spec value for the rest of the chip. (The rest of the chip may not function, but the oscillator should work.) It should also be verified that start-up occurs when the resonator has more than its upper tolerance limit of series resistance. (Put some resistance in series with the resonator for this test.) The bulk capacitors from XTAL1 and XTAL2 to ground should also be varied to their tolerance limits.

The same circuit, with appropriate changes in the software to lengthen the "on" time, can be used to test the steady-state characteristics of the oscillator, specifically the frequency, frequency stability, and amplitudes at XTAL1 and XTAL2.

As previously noted, the voltage swings at these pins are not critical, but they should be checked at the system's temperature limits to ensure that they are in good health. Observing these signals necessarily changes them somewhat. Observing the signal at XTAL2 requires that the capacitor at that pin be reduced to account for the oscilloscope probe capacitance. Observing the signal at XTAL1 requires the same consideration, plus a blocking capacitor (switch the oscilloscope input to AC), so as to not disturb the DC level at that pin. Alternatively, a MOSFET buffer such as the one shown in Figure 26 can be used. It should be verified by direct measurement that the ground clip on the scope probe is ohmically connected to the scope chassis (probes are incredibly fragile in this respect), and the observations should be made with the ground clip on the VSS pin, or very close to it. If the probe shield isn't operational and in use, the observations are worthless.

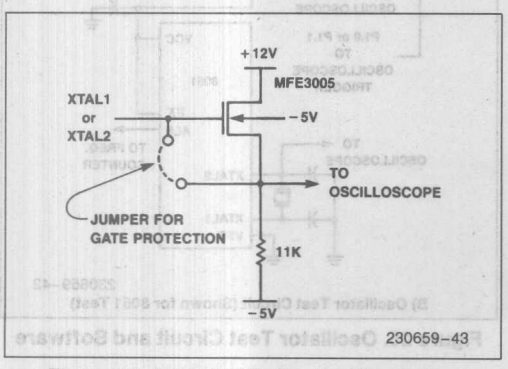


Figure 26. MOSFET Buffer for Observing Oscillator Signals

Frequency checks should be made with only the oscillator circuitry connected to XTAL1 and XTAL2. The ALE frequency can be counted, and the oscillator frequency derived from that. In systems where the frequency tolerance is only "nominal," the frequency should still be checked to ascertain that the oscillator isn't running in a spurious resonance or relaxation mode. Switching VCC off and on again repeatedly will help reveal a tendency to go into unwanted modes of oscillation.

The operation of the oscillator should then be verified under actual system running conditions. By this stage one will be able to have some confidence that the basic selection of components for the oscillator itself is suitable, so if the oscillator appears to malfunction in the system the fault is not in the selection of these components.

Troubleshooting Oscillator Problems

The first thing to consider in case of difficulty is that between the test jig and the actual application there may be significant differences in stray capacitances, particularly if the actual application is on a multi-layer board.

Noise glitches, that aren't present in the test jig but are in the application board, are another possibility. Capacitive coupling between the oscillator circuitry and other signal has already been mentioned as a source of miscounts in the internal clocking circuitry. Inductive coupling is also possible, if there are strong currents nearby. These problems are a function of the PCB layout.

Surrounding the oscillator components with "quiet" traces (VCC and ground, for example) will alleviate capacitive coupling to signals that have fast transition times. To minimize inductive coupling, the PCB layout should minimize the areas of the loops formed by the oscillator components. These are the loops that should be checked:

XTAL1 through the resonator to XTAL2;
XTAL1 through C_{X1} to the VSS pin;
XTAL2 through C_{X2} to the VSS pin.

It is not unusual to find that the grounded ends of C_{X1} and C_{X2} eventually connect up to the VSS pin only after looping around the farthest ends of the board. Not good.

Finally, it should not be overlooked that software problems sometimes imitate the symptoms of a slow-starting oscillator or incorrect frequency. Never underestimate the perversity of a software problem.

REFERENCES

1. Frerking, M. E., *Crystal Oscillator Design and Temperature Compensation,* Van Nostrand Reinhold, 1978.

2. Bottom, V., "The Crystal Unit as a Circuit Component," Ch. 7, *Introduction to Quartz Crystal Unit Design,* Van Nostrand Reinhold, 1982.

3. Parzen, B., *Design of Crystal and Other Harmonic Oscillators,* John Wiley & Sons, 1983.

4. Holmbeck, J. D., "Frequency Tolerance Limitations with Logic Gate Clock Oscillators, *31st Annual Frequency Control Symposium,* June, 1977.

5. Roberge, J. K., "Nonlinear Systems," Ch. 6, *Operational Amplifiers: Theory and Practice,* Wiley, 1975.

6. Eaton, S. S. *Timekeeping Advances Through COS/MOS Technology,* RCA Application Note ICAN-6086.

7. Eaton, S. S., *Micropower Crystal-Controlled Oscillator Design Using RCA COS/MOS Inverters,* RCA Application Note ICAN-6539.

8. Fisher, J. B., *Crystal Specifications for the Intel 8031/8051/8751 Microcontrollers,* Standard Crystal Corp. Design Data Note #2F.

9. Murata Mfg. Co., Ltd., *Ceramic Resonator "Ceralock" Application Manual.*

10. Kyoto Ceramic Co., Ltd., *Adaptability Test Between Intel 8049H and Kyocera Ceramic Resonators.*

11. Kyoto Ceramic Co., Ltd., *Technical Data on Ceramic Resonator Model KBR-6.0M, KBR-8.0M, KBR-11.0M Application for 8051 (Intel).*

12. NTK Technical Ceramic Division, NGK Spark Plug Co., Ltd., *NTKK Ceramic Resonator Manual.*

APPENDIX A
QUARTZ AND CERAMIC RESONATOR FORMULAS

Based on the equivalent circuit of the crystal, the impedance of the crystal is

$$Z_{XTAL} = \frac{(R_1 + j\omega L_1 + 1/j\omega C_1)\,(1/j\omega C_0)}{R_1 + j\omega L_1 + 1/j\omega C_1 + 1/j\omega C_0}$$

After some algebraic manipulation, this calculation can be written in the form

$$Z_{XTAL} = \frac{1}{j\omega(C_1 + C_0)} \bullet \frac{1 - \omega^2 L_1 C_1 + j\omega R_1 C_1}{1 - \omega^2 L_1 C_T + j\omega R_1 C_T}$$

where C_T is the capacitance of C_1 in series with C_0:

$$C_T = \frac{C_1 C_0}{C_1 + C_0}$$

The impedance of the crystal in parallel with an external load capacitance C_L is the same expression, but with $C_0 + C_L$ substituted for C_0:

$$Z_{XTAL}\,\|_{CL} = \frac{1}{j\omega(C_1 + C_0 + C_L)} \bullet \frac{1 - \omega^2 L_1 C_1 + j\omega R_1 C_1}{1 - \omega^2 L_1 C'_T + j\omega R_1 C'_T}$$

where C'_T is the capacitance of C_1 in series with $(C_0 + C_L)$:

$$C'_T = \frac{C_1(C_0 + C_L)}{C_1 + C_0 + C_L}$$

The impedance of the crystal in *series* with the load capacitance is

$$Z_{XTAL + CL} = Z_{XTAL} + \frac{1}{j\omega C_L}$$

$$= \frac{C_L + C_1 + C_0}{j\omega C_L (C_1 + C_0)} \bullet \frac{1 - \omega^2 L_1 C'_T + j\omega R_1 C'_T}{1 - \omega^2 L_1 C_T + j\omega R_1 C_T}$$

where C_T and C'_T are as defined above.

The phase angles of these impedances are readily obtained from the impedance expressions themselves:

$$\theta_{XTAL} = \arctan \frac{\omega R_1 C_1}{1 - \omega^2 L_1 C_1}$$

$$- \arctan \frac{\omega R_1 C_T}{1 - \omega^2 L_1 C_T} - \frac{\pi}{2}$$

$$\theta_{XTAL}\|_{C_L} = \arctan \frac{\omega R_1 C_1}{1 - \omega^2 L_1 C_1}$$

$$- \arctan \frac{\omega R_1 C'_T}{1 - \omega^2 L_1 C'_T} - \frac{\pi}{2}$$

$$\theta_{XTAL + C_L} = \arctan \frac{\omega R_1 C'_T}{1 - \omega^2 L_1 C'_T}$$

$$- \arctan \frac{\omega R_1 C_T}{1 - \omega^2 L_1 C_T} - \frac{\pi}{2}$$

The resonant ("series resonant") frequency is the frequency at which the phase angle is zero and the impedance is low. The antiresonant ("parallel resonant") frequency is the frequency at which the phase angle is zero and the impedance is high.

Each of the above θ-expressions contains two arctan functions. Setting the denominator of the argument of the first arctan function to zero gives (approximately) the "series resonant" frequency for that configuration. Setting the denominator of the argument of the second arctan function to zero gives (approximately) the "parallel resonant" frequency for that configuration.

For example, the resonant frequency of the crystal is the frequency at which

$$1 - \omega^2 L_1 C_1 = 0$$

Thus

$$\omega_s = \frac{1}{\sqrt{L_1 C_1}}$$

or

$$f_s = \frac{1}{2\pi\sqrt{L_1 C_1}}$$

It will be noted that the series resonant frequency of the "XTAL + CL" configuration (crystal in series with CL) is the same as the parallel resonant frequency of the "XTAL‖CL" configuration (crystal in parallel with C_L). This is the frequency at which

$$1 - \omega^2 L_1 C'_T = 0$$

Thus

$$\omega_a = \frac{1}{\sqrt{L_1 C'_T}}$$

or

$$f_a = \frac{1}{2\pi\sqrt{L_1 C'_T}}$$

This fact is used by crystal manufacturers in the process of calibrating a crystal to a specified load capacitance.

By subtracting the resonant frequency of the crystal from its antiresonant frequency, one can calculate the range of frequencies over which the crystal reactance is positive:

$$f_a - f_s = f_s(\sqrt{1 + C_1/C_0} - 1$$

$$f_s\left(\frac{C_1}{2C_0}\right)$$

Given typical values for C_1 and C_0, this range can hardly exceed 0.5% of fs. Unless the inverting amplifier in the positive reactance oscillator is doing something very strange indeed, the oscillation frequency is bound to be accurate to that percentage whether the crystal was calibrated for series operation or to any unspecified load capacitance.

Equivalent Series Resistance

ESR is the real part of Z_{XTAL} at the oscillation frequency. The oscillation frequency is the parallel resonant frequency of the "XTAL‖CL" configuration (which is the same as the series resonant frequency of the "XTAL + CL" configuration). Substituting this frequency into the Z_{XTAL} expression yields, after some algebraic manipulation,

$$ESR = \frac{R_1\left(\frac{C_0 + C_L}{C_L}\right)^2}{1 + \omega^2 C_1^2\left(\frac{C_0 + C_L}{C_L}\right)^2}$$

$$\cong R_1\left(1 + \frac{C_0}{C_L}\right)^2$$

Drive Level

The power dissipated by the crystal is $I_1^2 R_1$, where I_1 is the RMS current in the motional arm of the crystal. This current is given by $V_x/|Z_1|$, where V_x is the RMS voltage across the crystal, and $|Z_1|$ is the magnitude of the impedance of the motional arm. At the oscillation frequency, the motional arm is a positive (inductive) reactance in parallel resonance with $(C_0 + C_L)$. Therefore $|Z_1|$ is approximately equal to the magnitude of the reactance of $(C_0 + C_L)$:

$$|Z_1| = \frac{1}{2\pi f(C_0 + C_L)}$$

where f is the oscillation frequency. Then,

$$P = I_1^2 R_1 = \left(\frac{V_x}{|Z_1|}\right)^2 R_1$$

$$= [2\pi f\,(C_0 + C_L)\,V_x]^2\,R_1$$

The waveform of the voltage across the crystal (XTAL1 to XTAL2) is approximately sinusoidal. If its peak value is VCC, then V_x is $VCC/\sqrt{2}$. Therefore,

$$P = 2R_1\,[\pi f\,(C_0 + C_L)\,VCC]^2$$

APPENDIX B
OSCILLATOR ANALYSIS PROGRAM

The program is written in BASIC. BASIC is excruciatingly slow, but it has some advantages. For one thing, more people know BASIC than FORTRAN. In addition, a BASIC program is easy to develop, modify, and "fiddle around" with. Another important advantage is that a BASIC program can run on practically any small computer system.

Its slowness is a problem, however. For example, the routine which calculates the "start-up time constant" discussed in the text may take several hours to complete. A person who finds this program useful may prefer to convert it to FORTAN, if the facilities are available.

Limitations of the Program

The program was developed with specific reference to 8051-type oscillator circuitry. That means the on-chip amplifier is a simple inverter, and not a Schmitt Trigger. The 8096, the 80C51, the 80C48 and 80C49 all have simple inverters. The 8096 oscillator is almost identical to the 8051, differing mainly in the input protection circuitry. The CHMOS amplifiers have somewhat different parameters (higher gain, for example), and different transition levels than the 8051.

The MCS-48 family is specifically included in the program only to the extent that the input-output curve used in the steady-state analysis is that of a Schmitt Trigger, if the user identifies the device under analysis as an MCS-48 device. The analysis does not include the voltage dependent phase shift of the Schmitt Trigger.

The clamping action of the input protection circuitry is important in determining the steady-state amplitudes. The steady-state routine accounts for it by setting the negative peak of the XTAL1 signal at a level which depends on the amplitude of the XTAL1 signal in accordance with experimental observations. It's an exercise in curve-fitting. A user may find a different type of curve works better. Later steppings of the chips may behave differently in this respect, having somewhat different types of input protection circuitry.

It should be noted that the analysis ignores a number of important items, such as high-frequency effects in the on-chip circuitry. These effects are difficult to predict, and are no doubt dependent on temperature, frequency, and device sample. However, they can be simulated to a reasonable degree by adding an "output capacitance" of about 20 pF to the circuit model (i.e., in parallel with CX2) as described below.

Notes on Using the Program

The program asks the user to input values for various circuit parameters. First the crystal (or ceramic resonator) parameters are asked for. These are R1, L1, C1, and C0. The manufacturer can supply these values for selected samples. To obtain any kind of correlation between calculation and experiment, the values of these parameters must be known for the specific sample in the test circuit. The value that should be entered for C0 is the C0 of the crystal itself plus an estimated 7 pF to account for the XTAL1-to-XTAL2 pin capacitance, plus any other stray capacitance paralleling the crystal that the user may feel is significant enough to be included.

Then the program asks for the values of the XTAL1-to-ground and XTAL2-to-ground capacitances. For CXTAL1, enter the value of the externally connected bulk capacitor plus an estimated 7 pF for pin capacitance. For CXTAL2, enter the value of the externally connected bulk capacitor plus an estimated 7 pF for pin capacitance plus about 20 pF to simulate high-frequency roll-off and phase shifts in the on-chip circuitry.

Next the program asks for values for the small-signal parameters of the on-chip amplifier. Typically, for the 8051/8751,

Amplifier Gain Magnitude	= 15
Feedback Resistance	= 2300 KΩ
Output Resistance	= 2 KΩ

The same values can be used for MCS-48 (NMOS and HMOS) devices, but they are difficult to verify, because the Schmitt Trigger does not lend itself to small-signal measurements.

```
100 DEFDBL C,D,F,G,L,P,R,S,X
200 REM                                                          APRIL 8, 1983
300 REM ************************************************************************
400 REM
500 REM                          FUNCTIONS
600 REM
700 REM
800 REM FNZM(R,X) = MAGNITUDE OF A COMPLEX NUMBER, |R+jX|
900 DEF FNZM(R,X) = SQR(R^2+X^2)
1000 REM
1100 REM FNZP(R,X) = ANGLE OF A COMPLEX NUMBER
1200 REM           = 180/PI*ARCTAN(X/R)          IF R>0
1300 REM           = 180/PI*ARCTAN(X/R) + 180    IF R<0 AND X>0
1400 REM           = 180/PI*ARCTAN(X/R) - 180    IF R<0 AND X<0
1500 DEF FNZP(R,X) = 180/PI*ATN(X/R) - (SGN(R)-1)*SGN(X)*90
1600 REM
1700 REM    INDUCTIVE IMPEDANCE AT COMPLEX FREQUENCY S+jF (HZ)
1800 REM          Z = 2*PI*S*L  + j2*PI*F*L
1900 REM            = FNRL(S,L) + jFNXL(F,L)
2000 DEF FNRL(SL,LL) = 2##PI*SL*LL
2100 DEF FNXL(FL,LL) = 2##PI*FL*LL
2200 REM
2300 REM    CAPACITIVE IMPEDANCE AT COMPLEX FREQUENCY S+jF (HZ)
2400 REM          Z = 1/[2*PI*(S+jF)*C]
2500 REM            = S/[2*PI*(S^2+F^2)*C] + j(-F)/[2*PI*(S^2+F^2)C]
2600 REM            = FNRC(S,F,C) + jFNXC(S,F,C)
2700 DEF FNRC(SC,FC,CC) = SC/(2##PI*(SC^2+FC^2)*CC)
2800 DEF FNXC(SC,FC,CC) = -FC/(2##PI*(SC^2+FC^2)*CC)
2900 REM
3000 REM    RATIO OF TWO COMPLEX NUMBERS
3100 REM          RA+jXA      RA*RB+XA*XB       XA*RB-RA*XB
3200 REM          ------  =  -----------  + j  -----------
3300 REM          RB+jX3       RB^2+XB^2         RB^2+XB^2
3400 REM                  = FNRR(RA,XA,RB,XB) + jFNXR(RA,XA,RB,XB)
3500 DEF FNRR(RA,XA,RB,XB) = (RA*RB+XA*XB)/(RB^2+XB^2)
3600 DEF FNXR(RA,XA,RB,XB) = (XA*RB-XB*RA)/(RB^2+XB^2)
3700 REM
3800 REM    PRODUCT OF TWO COMPLEX NUMBERS
3900 REM        (RA+jXA)*(RB+jXB)  =  RA*RB-XA*XB + j(XA*RB+RA*XB)
4000 REM                           = FNRM(RA,XA,RB,XB) + jFNXM(RA,XA,RB,XB)
4100 DEF FNRM(RA,XA,RB,XB) = RA*RB - XA*XB
4200 DEF FNXM(RA,XA,RB,XB) = RA*XB + RB*XA
4300 REM
4400 REM
4500 REM    PARALLEL IMPEDANCES
4600 REM                                (RA+jXA)*(RB+jXB)
4700 REM    (RA+jXA)||(RB+jXB)  =  ------------------
4800 REM                             RA+RB +j(XA+XB)
4900 REM
5000 REM      RA*(RB^2+XB^2)+RB*(RA^2+XA^2)       XA*(RB^2+XB^2)+XB*(RA^2+XA^2)
5100 REM   =  ------------------------------  +  j ------------------------------
5200 REM          (RA+RB)^2 + (XA+XB)^2                (RA+RB)^2 + (XA+XB)^2
5300 REM
5400 REM   = FNRP(RA,XA,RB,XB) + jFNXP(RA,XA,RB,XB)
5500 DEF FNRP(RA,XA,RB,XB) = (RA*(RB^2+XB^2) + RB*(RA^2+XA^2))/((RA+RB)^2 + (XA+XB)^2)
5600 DEF FNXP(RA,XA,RB,XB) = (XA*(RB^2+XB^2) + XB*(RA^2+XA^2))/((RA+RB)^2 + (XA+XB)^2)
5700 REM
5800 REM ************************************************************************
5900 REM
6000 REM                BEGIN COMPUTATIONS
6100 REM
6200 LET PI = 3.141592654#
6300 REM
6400 REM             DEFINE CIRCUIT PARAMETERS
6500 GOSUB 14500
6600 REM
6700 REM  ESTABLISH NOMINAL RESONANT AND ANTIRESONANT CRYSTAL FREQUENCIES
6800 FS = FIX(1/(2*PI*SQR(L1*C1)))
6900 FA = FIX(1/(2*PI*SQR(L1*C1*C0/(C1+C0))))
7000 PRINT
7100 PRINT "XTAL IS SERIES RESONANT AT ";FS;" HZ"
7200 PRINT "       PARALLEL RESONANT AT ";FA;" HZ"
7300 PRINT
7400 PRINT "SELECT  1. LIST PARAMETERS"
7500 PRINT "        2. CIRCUIT ANALYSIS"
7600 PRINT "        3. OSCILLATION FREQUENCY"
7700 PRINT "        4. START-UP TIME CONSTANT"
7800 PRINT "        5. STEADY-STATE ANALYSIS"
```

230659-44

```
7900 PRINT
8000 INPUT N
8100 IF N=1 THEN PRINT ELSE 8600
8200 REM
8300 REM ---------- LIST PARAMETERS ----------
8400 GOSUB 17100
8500 GOTO 6800
8600 IF N=2 THEN PRINT ELSE 9400
8700 REM
8800 REM ---------- CIRCUIT ANALYSIS ----------
8900 PRINT " FREQUENCY S+JF  TYPE (S),(F) "
9000 INPUT SQ,FQ
9100 GOSUB 20200
9200 GOSUB 26600
9300 GOTO 6800
9400 IF N=3 THEN 10300 ELSE 11000
9500 REM
9600 REM ---------- OSCILLATION FREQUENCY ----------
9700 CL = CX*CY/(CX+CY) + CO
9800 FQ = FIX(1/(2*PI*SQR(L1*C1*CL/(C1+CL))))
9900 SQ = 0
10000 DF = FIX(10^INT(LOG(FA-FS)/LOG(10)-2)+.5)
10100 DS = 0
10200 RETURN
10300 GOSUB 9700
10400 GOSUB 30300
10500 PRINT
10600 PRINT
10700 PRINT "FREQUENCY AT WHICH LOOP GAIN HAS ZERO PHASE ANGLE."
10800 GOSUB 26600
10900 GOTO 6800
11000 IF N=4 THEN PRINT ELSE 12200
11100 REM
11200 REM ---------- START-UP TIME CONSTANT ----------
11300 PRINT "THIS WILL TAKE SOME TIME "
11400 GOSUB 9700
11500 GOSUB 37700
11600 PRINT
11700 PRINT
11800 PRINT "FREQUENCY AT WHICH LOOP GAIN = 1 AT 0 DEGREES."
11900 GOSUB 26600
12000 PRINT : PRINT "THIS YIELDS A START-UP TIME CONSTANT OF ";CSNG(1000000!/(2*PI*SQ));" MICROSECS"
12100 GOTO 6800
12200 IF N=5 THEN PRINT ELSE 7300
12300 REM
12400 REM ---------- STEADY-STATE ANALYSIS ----------
12500 PRINT "STEADY-STATE ANALYSIS"
12600 PRINT
12700 PRINT "SELECT:  1. 8031/8051"
12800 PRINT "         2. 8751"
12900 PRINT "         3. 8035/8039/8040/8048/8049"
13000 PRINT "         4. 8748/8749"
13100 INPUT IC%
13200 IF IC%<1 OR IC%>4 THEN 12600
13300 GOSUB 46900
13400 GOTO 7300
13500 REM   SUBROUTINE BELOW DEFINES INPUT-OUTPUT CURVE OF OSCILLATOR CKT
13600 IF IC%>2 AND VO=5 AND VI<2 THEN RETURN
13700 VO = -10*VI + 15
13800 IF VO>5 THEN VO = 5
13900 IF VO< 2 THEN VO = 2
14000 IF IC%>2 AND VO>2 THEN VO = 5
14100 RETURN
14200 REM
14300 REM ****************************************************************
14400 REM
14500 REM              DEFINE CIRCUIT PARAMETERS
14600 REM
14700 INPUT " R1 (OHMS)";R1
14800 INPUT " L1 (HENRY)";L1
14900 INPUT " C1 (PF)";X
15000 C1 = X*1E-12
15100 INPUT " CO (PF)",X
15200 CO = X*1E-12
15300 INPUT " CXTAL1 (PF)";X
15400 CX = X*1E-12
15500 INPUT " CXTAL2 (PF)";X
15600 CY = X*1E-12
```

230659-45

```
15700 INPUT " GAIN FACTOR MAGNITUDE";AV#
15800 INPUT " AMP FEEDBACK RESISTANCE (K-OHMS)",X
15900 RX = X*1000#
16000 INPUT " AMP OUTPUT RESISTANCE (K-OHMS)";X
16100 RO = X*1000#
16200 REM
16300 REM
16400 REM          LIST CURRENT PARAMETER VALUES
16500 GOSUB 17100
16600 RETURN
16700 REM
16800 REM
16900 REM ************************************************************
17000 REM
17100 REM              LIST CURRENT PARAMETER VALUES
17200 REM
17300 PRINT
17400 PRINT "CURRENT PARAMETER VALUES  1  R1 = ";R1," OHMS"
17500 PRINT "                         2  L1 = ";CSNG(L1)," HENRY"
17600 PRINT "                         3  C1 = ";CSNG(C1*1E+12)," PF"
17700 PRINT "                         4  CO = ";CSNG(CO*1E+12)," PF"
17800 PRINT "                         5  CXTAL1 = ";CSNG(CX*1E+12)," PF"
17900 PRINT "                         6  CXTAL2 = ";CSNG(CY*1E+12)," PF"
18000 PRINT "    7  AMPLIFIER GAIN MAGNITUDE = ";AV#
18100 PRINT "    8       FEEDBACK RESISTANCE = ";CSNG(RX*.001)," K-OHMS"
18200 PRINT "    9         OUTPUT RESISTANCE = ";CSNG(RO*.001)," K-OHMS"
18300 PRINT
18400 PRINT "TO CHANGE A PARAMETER VALUE, TYPE (PARAM NO.),(NEW VALUE)."
18500 PRINT "OTHERWISE, TYPE 0,0 "
18600 INPUT N%,X
18700 IF N%=0 THEN RETURN
18800 IF N%=1 THEN R1 = X
18900 IF N%=2 THEN L1 = X
19000 IF N%=3 THEN C1 = X*1E-12
19100 IF N%=4 THEN CO = X*1E-12
19200 IF N%=5 THEN CX = X*1E-12
19300 IF N%=6 THEN CY = X*1E-12
19400 IF N%=7 THEN AV# = X
19500 IF N%=8 THEN RX = X*1000!
19600 IF N%=9 THEN RO = X*1000!
19700 GOTO 17400
19800 REM
19900 REM
20000 REM ************************************************************
20100 REM
20200 REM                  CIRCUIT ANALYSIS
20300 REM
20400 REM  This routine calculates the loop gain at complex frequency SQ+jFQ.
20500 REM
20600 REM  1. Crystal impedance  RE + jXE
20700 REM
20800 X1 = FNXL(FQ,L1) + FNXC(SQ,FQ,C1)
20900 RE = FNRP((R1+FNRL(SQ,L1)+FNRC(SQ,FQ,C1)),X1,FNRC(SQ,FQ,CO),FNXC(SQ,FQ,CO))
21000 XE = FNXP((R1+FNRL(SQ,L1)+FNRC(SQ,FQ,C1)),X1,FNRC(SQ,FQ,CO),FNXC(SQ,FQ,CO))
21100 REM
21200 REM  2. RF + jXF  =  (RE+jXE)||(amplifier feedback resistance)
21300 REM
21400 RF = FNRP(RX,0,RE,XE)
21500 XF = FNXP(RX,0,RE,XE)
21600 REM
21700 REM  3. Input impedance  Z1 = RI + jXI = impedance of CXTAL1
21800 REM
21900 RI = FNRC(SQ,FQ,CX)
22000 XI = FNXC(SQ,FQ,CX)
22100 REM
22200 REM  4. Load impedance  ZL = (impedance of CXTAL2)||[(RF+RI)+j(XF+XI)]
22300 REM
22400 RL = FNRP((RF+RI),(XF+XI),FNRC(SQ,FQ,CY),FNXC(SQ,FQ,CY))
22500 XL = FNXP((RF+RI),(XF+XI),FNRC(SQ,FQ,CY),FNXC(SQ,FQ,CY))
22600 REM
22700 REM  5. Amplifier gain  A = -AV*ZL/(ZL+RO)
22800 REM                       = A(real) + jA(imaginary)
22900 REM
23000 AR# = -AV#*FNRR(RL,XL,(RO+RL),XL)
23100 AI# = -AV#*FNXR(RL,XL,(RO+RL),XL)
23200 REM
23300 REM  6. Feedback ratio  (beta) = (RI+jXI)/[(RF+RI)+j(XF+XI)]
23400 REM                           = B(real) + jB(imaginary)
```

230659-46

```
23500 REM
23600 BR# = FNRR(RI,XI,(RI+RF),(XI+XF))
23700 BI# = FNXR(RI,XI,(RI+RF),(XI+XF))
23800 REM
23900 REM  7. Amplifier gain in magnitude/phase form  AR+jAI = A at AP degrees
24000 REM
24100 A = FNZM(AR#,AI#)
24200 AP = FNZP(AR#,AI#)
24300 REM
24400 REM  8  (beta) in magnitude/phase form  BR+jBI = B at BP degrees
24500 REM
24600 B = FNZM(BR#,BI#)
24700 BP = FNZP(BR#,BI#)
24800 REM
24900 REM  9  Loop gain  G = (BR+jBI)*(AR+jAI)
25000 REM                 = G(real) + jG(imaginary)
25100 REM
25200 GR = FNRM(AR#,AI#,BR#,BI#)
25300 GI = FNXM(AR#,AI#,BR#,BI#)
25400 REM
25500 REM  10  Loop gain in magnitude/phase form  GR+jGI = AL at AQ degrees
25600 REM
25700 AL = FNZM(GR,GI)
25800 AQ = FNZP(GR,GI)
25900 RETURN
26000 REM
26100 REM
26200 REM ************************************************************
26300 REM
26400 REM              PRINT CIRCUIT ANALYSIS RESULTS
26500 REM
26600 PRINT
26700 PRINT " FREQUENCY = ";SQ;" + J";FQ;" HZ"
26800 PRINT " XTAL IMPEDANCE = ";FNZM(RE,XE);" OHMS AT ";FNZP(RE,XE);" DEGREES"
26900 PRINT "         (RE = ";CSNG(RE);" OHMS)"
27000 PRINT "         (XE = ";CSNG(XE);" OHMS)"
27100 PRINT " LOAD IMPEDANCE = ";FNZM(RL,XL);" OHMS AT ";FNZP(RL,XL);" DEGREES"
27200 PRINT " AMPLIFIER GAIN = ";A;" AT ";AP;" DEGREES"
27300 PRINT " FEEDBACK RATIO = ";B;" AT ";BP;" DEGREES"
27400 PRINT " LOOP GAIN = ";AL;" AT ";AQ;" DEGREES"
27500 RETURN
27600 REM
27700 REM
27800 REM ************************************************************
27900 REM
28000 REM                SEARCH FOR FREQUENCY (S+JF)
28100 REM          AT WHICH LOOP GAIN HAS ZERO PHASE ANGLE
28200 REM
28300 REM  This routine searches for the frequency at which the imaginary part
28400 REM  of the loop gain is zero.  The algorithm is as follows:
28500 REM    1. Calculate the sign of the imaginary part of the loop gain (GI).
28600 REM    2. Increment the frequency.
28700 REM    3. Calculate the sign of GI at the incremented frequency.
28800 REM    4. If the sign of GI has not changed, go back to 2.
28900 REM    5. If the sign of GI has changed, and this frequency is within
29000 REM       1Hz of the previous sign-change, exit the routine.
29100 REM    6. Otherwise, divide the frequency increment by -10.
29200 REM    7. Go back to 2.
29300 REM  The routine is entered with the starting frequency SQ+jFQ and
29400 REM  starting increment DS+jDF already defined by the calling program.
29500 REM  In actual use either DS or DF is zero, so the routine searches for
29600 REM  a GI=0 point by incrementing either SQ or FQ while holding the other
29700 REM  constant.  It returns control to the calling program with the
29800 REM  incremented part of the frequency being within 1Hz of the actual
29900 REM  GI=0 point.
30000 REM
30100 REM  1. CALCULATE THE SIGN OF THE IMAGINARY PART OF THE LOOP GAIN (GI).
30200 REM
30300 GOSUB 20200
30400 GOSUB 26600
30500 IF GI=0 THEN RETURN
30600 SX% = INT(SGN(GI))
30700 IF SX%=+1 THEN DS = -DS
30800 REM  (REVERSAL OF DS FOR GI=0 IS FOR THE POLE-SEARCH ROUTINE.)
30900 REM
31000 REM  2  INCREMENT THE FREQUENCY.
31100 REM
31200 SP = SQ
```

230659-47

```
31300 FP = FQ
31400 SQ = SQ + DS
31500 FQ = FQ + DF
31600 REM
31700 REM  3. CALCULATE THE SIGN OF GI AT THE INCREMENTED FREQUENCY
31800 REM
31900 GOSUB 20200
32000 GOSUB 26600
32100 IF INT(SGN(GI))=0 THEN RETURN
32200 REM
32300 REM  4. IF THE SIGN OF GI HAS NOT CHANGED, GO BACK TO 2
32400 REM
32500 IF SX%+INT(SGN(GI))=0 THEN PRINT ELSE 31400
32600 SX% = -SX%
32700 REM
32800 REM  5. IF THE SIGN OF GI HAS CHANGED, AND IF THIS FREQUENCY IS WITHIN
32900 REM     1HZ OF THE PREVIOUS SIGN-CHANGE, AND IF GI IS NEGATIVE, THEN
33000 REM     EXIT THE ROUTINE. (THE ADDITIONAL REQUIREMENT FOR NEGATIVE GI
33100 REM     IS FOR THE POLE-SEARCH ROUTINE.)
33200 REM
33300 IF ABS(SP-SQ)<1 AND ABS(FP-FQ)<1 AND SX%=-1 THEN RETURN
33400 REM
33500 REM  6. DIVIDE THE FREQUENCY INCREMENT BY -10
33600 REM
33700 DS = -DS/10#
33800 DF = -DF/10#
33900 REM
34000 REM  7. GO BACK TO 2
34100 REM
34200 GOTO 31200
34300 REM
34400 REM
34500 REM ****************************************************************
34600 REM
34700 REM                SEARCH FOR POLE FREQUENCY
34800 REM
34900 REM  This routine searches for the frequency at which the loop gain = 1
35000 REM  at 0 degrees.  That frequency is the pole frequency of the closed-
35100 REM  loop gain function.  The pole frequency is a complex number, SQ+jFQ
35200 REM  (Hz).  Oscillator start-up ensues if SQ>0.  The algorithm is based on
35300 REM  the calculated behavior of the phase angle of the loop gain in the
35400 REM  region of interest on the complex plane.  The locus of points of zero
35500 REM  phase angle crosses the j-axis at the oscillation frequency and at
35600 REM  some higher frequency.  In between these two crossings of the j-axis,
35700 REM  the locus lies in Quadrant I of the complex plane, forming an
35800 REM  approximate parabola which opens to the left.  The basic plan is to
35900 REM  follow the locus from where it crosses the j-axis at the oscillation
36000 REM  frequency, into Quadrant I, and find the point on that locus where
36100 REM  the loop gain has a magnitude of 1.  The algorithm is as follows:
36200 REM     1. Find the oscillation frequency, 0+jFQ.
36300 REM     2. At this frequency calculate the sign of (AL-1).  (AL = magnitude
36400 REM        of loop gain.)
36500 REM     3. Increment FQ.
36600 REM     4. For this value of FQ, find the value of SQ for which the loop
36700 REM        gain has zero phase.
36800 REM     5. For this value of SQ+jFQ, calculate the sign of (AL-1).
36900 REM     6. If the sign of (AL-1) has not changed, go back to 3.
37000 REM     7. If the sign of (AL-1) has changed, and this value of FQ is
37100 REM        within 1Hz of the previous sign-change, exit the routine.
37200 REM     8. Otherwise, divide the FQ-increment by -10.
37300 REM     9. Go back to 3.
37400 REM
37500 REM  1. FIND THE OSCILLATION FREQUENCY, 0+jFQ.
37600 REM
37700 GOSUB 9700
37800 GOSUB 30300
37900 REM
38000 REM  2. AT THIS FREQUENCY, CALCULATE THE SIGN OF (AL-1).
38100 REM
38200 SY% = INT(SGN(AL-1!))
38300 IF SY%=-1 THEN STOP
38400 REM  ESTABLISH INITIAL INCREMENTATION VALUE FOR FQ.
38500 F1 = FQ
38600 DF = (FA-F1)/10#
38700 GOSUB 30300
38800 DE = (FQ-F1)/10#
38900 DF = 0
39000 FQ = F1
```

230659-48

```
39100 REM
39200 REM  3. INCREMENT FQ
39300 REM
39400 FQ = FQ + DE
39500 REM
39600 REM  4. FOR THIS VALUE OF FQ, FIND THE VALUE OF SQ FOR WHICH THE LOOP
39700 REM     GAIN HAS ZERO PHASE. (THE ROUTINE WHICH DOES THAT NEEDS DF = 0,
39800 REM     SO THAT IT CAN HOLD FQ CONSTANT, AND NEEDS AN INITIAL VALUE FOR
39900 REM     DS, WHICH IS ARBITRARILY SET TO DS = 1000.)
40000 REM
40100 DS = 1000#
40200 SQ = 0
40300 GOSUB 30300
40400 IF AL=1! THEN RETURN
40500 REM
40600 REM  5. FOR THIS VALUE OF SQ+jFQ, CALCULATE THE SIGN OF (AL-1).
40700 REM  6. IF THE SIGN OF (AL-1) HAS NOT CHANGED, GO BACK TO 3.
40800 REM
40900 IF SY%+INT(SGN(AL-1!))=0 THEN PRINT ELSE 39400
41000 REM
41100 REM  7. IF THE SIGN OF (AL-1) HAS CHANGED, AND THIS VALUE OF FQ IS WITHIN
41200 REM     1HZ OF THE PREVIOUS SIGN-CHANGE, EXIT THE ROUTINE.
41300 REM
41400 IF ABS(F1-FQ)<1 THEN RETURN
41500 REM
41600 REM  8. DIVIDE THE FQ-INCREMENT BY -10.
41700 REM
41800 DE = -DE/10#
41900 F1 = FQ
42000 SY% = -SY%
42100 REM
42200 REM  9. GO BACK TO 3.
42300 REM
42400 GOTO 39400
42500 REM
42600 REM
42700 REM ************************************************************
42800 REM
42900 REM                  STEADY-STATE ANALYSIS
43000 REM
43100 REM  The circuit model used in this analysis is similar to the one used
43200 REM  in the small-signal analysis, but differs from it in two respects.
43300 REM  First, it includes clamping and clipping effects described in the
43400 REM  text. Second, the voltage source in the Thevenin equivalent of the
43500 REM  amplifier is controlled by the input voltage in accordance with an
43600 REM  input-output curve defined elsewhere in the program.
43700 REM    The analysis applies a sinusoidal input signal of arbitrary
43800 REM  amplitude, at the oscillation frequency, to the XTAL1 pin, then
43900 REM  calculates the resulting waveform from the voltage source. Using
44000 REM  standard Fourier techniques, the fundamental frequency component of
44100 REM  this waveform is extracted. This frequency component is then
44200 REM  multiplied by the factor |ZL/(ZL+RO)|, and the result is taken to be
44300 REM  the signal appearing at the XTAL2 pin. This signal is then
44400 REM  multiplied by the feedback ratio (beta), and the result is taken to
44500 REM  be the signal appearing at the XTAL1 pin. The algorithm is now
44600 REM  repeated using this computed XTAL1 signal as the assumed input
44700 REM  sinusoid. Every time the algorithm is repeated, new values appear at
44800 REM  XTAL1 and XTAL2, but the values change less and less with each
44900 REM  repetition. Eventually they stop changing. This is the steady-state.
45000 REM    The algorithm is as follows:
45100 REM  1. Compute approximate oscillation frequency.
45200 REM  2. Call a circuit analysis at this frequency.
45300 REM  3. Find the quiescent levels at XTAL1 and XTAL2 (to establish the
45400 REM     beginning DC level at XTAL1).
45500 REM  4. Assume an initial amplitude for the XTAL1 signal.
45600 REM  5. Correct the DC level at XTAL1 for clamping effects, if necessary.
45700 REM  6. Using the appropriate input-output curve, extract a DC level and
45800 REM     the fundamental frequency component (multiplying the latter by
45900 REM     |ZL/(ZL+RO)|).
46000 REM  7. Clip off the negative portion of this output signal, if the
46100 REM     negative peak falls below zero.
46200 REM  8. If this signal, multiplied by (beta), differs from the input
46300 REM     amplitude by less than 1mV, or if the algorithm has been repeated
46400 REM     10 times, exit the routine.
46500 REM  9. Otherwise, multiply the XTAL2 amplitude by (beta) and feed it
46600 REM     back to XTAL1, and go back to 5.
46700 REM
46800 REM     1. COMPUTE APPROXIMATE OSCILLATION FREQUENCY.
```

230659-49

```
46900 GOSUB 9700
47000 REM
47100 REM     2. CALL A CIRCUIT ANALYSIS AT THIS FREQUENCY.
47200 GOSUB 20800
47300 PRINT : PRINT : PRINT "ASSUMED OSCILLATION FREQUENCY: "
47400 GOSUB 26600
47500 PRINT : PRINT
47600 REM
47700 REM     3. FIND QUIESCENT POINT
47800 REM  (At quiescence the voltages at XTAL1 and XTAL2 are equal. This
47900 REM   voltage level is found by trial-and-error, based on the input-
48000 REM   output curve, so that a person can change the input-output curve
48100 REM   as desired without having to re-calculate the quiescent point.)
48200 VI = 0
48300 VB = 1
48400 K1 = 1
48500 VI = VI + VB
48600 GOSUB 13600
48700 IF ABS(VO-VI)<.001 THEN 49200
48800 IF K1+SGN(VO-VI)=0 THEN 48900 ELSE 48500
48900 K1 = SGN(VO-VI)
49000 VB = -VB/10
49100 GOTO 48500
49200 VB = VI
49300 PRINT "QUIESCENT POINT = ";VB
49400 REM
49500 REM     4. ASSUME AN INITIAL AMPLITUDE FOR THE XTAL1 SIGNAL.
49600 EI = .01
49700 NR% = 0
49800 REM
49900 REM     5. CORRECT FOR CLAMPING EFFECTS, IF NECESSARY.
50000 REM  (K1 and K2 are curve-fitting parameters for the ROM parts.)
50100 K1 = (2.5-VB)/(3-VB)
50200 K2 = (VB-1.25)/(3-VB)
50300 IF IC%=2 OR IC%=4 THEN IF EI<(VB+.5) THEN EO = VB ELSE EO = EI - .5
50400 IF IC%=1 OR IC%=3 THEN IF EI<(VB+.5) THEN EO = VB ELSE EO = K1*EI+K2
50500 NR% = NR% + 1
50600 REM
50700 REM     6. DERIVE XTAL2 AMPLITUDE
50800 VO = 0
50900 VC = 0
51000 VS = 0
51100 FOR N% = -25 TO +24
51200 VI = EO - EI*COS(PI*N%/25)
51300 GOSUB 13600
51400 VO = VO + VO
51500 VC = VC + VO*COS(PI*N%/25)
51600 VS = VS + VO*SIN(PI*N%/25)
51700 NEXT N%
51800 VO = VO/50
51900 V1 = SQR(VC^2+VS^2)/25*FNZM(RL,XL)/FNZM((RL+RO),XL)
52000 REM
52100 REM     7. CLIP XTAL2 SIGNAL.
52200 IF VO-V1<0 THEN VL = 0 ELSE VL = VO-V1
52300 PRINT : PRINT "XTAL1 SWING = ";EO-EI;" TO ";EO+EI
52400 PRINT "XTAL2 SWING = ";VL;" TO ";VO+V1
52500 REM
52600 REM     8. TEST FOR TERMINATION.
52700 IF ABS(EI-V1*B)<.001 OR NR%=10 THEN RETURN
52800 REM
52900 REM     9. FEED BACK TO XTAL1 AND REPEAT
53000 EI = V1*B
53100 GOTO 50300
```

230659-50

intel

APPLICATION NOTE

AP-252

September 1987

Designing With The 80C51BH

TOM WILLIAMSON
MCO APPLICATIONS ENGINEER

Order Number: 270068-002

CMOS EVOLVES

The original CMOS logic families were the 4000-series and the 74C-series circuits. The 74C-series circuits are functional equivalents to the corresponding numbered 74-series TTL circuits, but have CMOS logic levels and retain the other well known characteristics of CMOS logic.

These characteristics are: low power consumption, high noise immunity, and slow speed. The low power consumption is inherent to the nature of the CMOS circuit. The noise immunity is due partly to the CMOS logic levels, and partly to the slowness of the circuits. The slow speed is due to the technology used to construct the transistors in the circuit.

The technology used is called metal-gate CMOS, because the transistor gates are formed by metal deposition. More importantly, the gates are formed after the drain and source regions have been defined, and must overlap the source and drain somewhat to allow for alignment tolerances. This overlap plus the relatively large size of the transistors themselves result in high electrode capacitance, and that is what limits the speed of the circuit.

High speed CMOS became feasible with the development of the self-aligning silicon gate technology. In this process polysilicon gates are deposited **before** the source and drain regions are defined. Then the source and drain regions are formed by ion implantation using the gate itself as a mask for the implantation. This eliminates most of the overlap capacitance. In addition, the process allows smaller transistors. The result is a significant increase in circuit speed. The 74HC-series of CMOS logic circuits is based on this technology, and has speeds comparable to LS TTL, which is to say about 10 times faster than the 74C-series circuits.

The size reduction that contributes to the higher speed also demands an accompanying reduction in the maximum supply voltage. High-speed CMOS is generally limited to 6V.

WHAT IS CHMOS?

CHMOS is the name given to Intel's high-speed CMOS processes. There are two CHMOS processes, one based on an n-well structure and one based on a p-well structure. In the n-well structure, n-type wells are diffused into a p-type substrate. Then the n-channel transistors (nFETs) are built into the substrate and pFETs are built into the n-wells. In the p-well structure, p-type wells are diffused into an n-type substrate. Then the nFETs are built into the wells and pFETs, into the substrate. Both processes have their advantages and disadvantages, which are largely transparent to the user.

Lower operating voltages are easier to obtain with the p-well structure than with the n-well structure. But the p-well structure does not easily adapt to an EPROM which would be pin-for-pin compatible with HMOS EPROMs. On the other hand the n-well structure can be based on the solidly founded HMOS process, in which nFETs are built into a p-type substrate. This allows somewhat more than half of the transistors in a CHMOS chip to be constructed by processes that are already well characterized.

Currently Intel's CHMOS microcontrollers and memory products are n-well devices, whereas CHMOS microprocessors are p-well devices.

Further discussion of the CHMOS technology is provided in References 1 and 2 (which are reprinted in the Microcontroller Handbook).

THE MCS®-51 FAMILY IN CHMOS

The 80C51BH is the CHMOS version of Intel's original 8051. The 80C31BH is the ROMless 80C51BH, equivalent to the 8031. These CHMOS devices are architecturally identical with their HMOS counterparts, except that they have two added features for reduced power. These are the Idle and Power Down modes of operation.

In most cases, an 80C51BH can directly replace the 8051 in existing applications. It can execute the same code at the same speed, accept signals from the same sources, and drive the same loads. However, the 80C51BH covers a wider range of speeds, will emit CMOS logic levels to CMOS loads, and will draw about 1/10 the current of an 8051 (and less yet in the reduced power modes). Interchangeability between the HMOS and CHMOS devices is discussed in more detail in the final section of this Application Note.

It should be noted that the 80C51BH CPU is not static. That means if the clock frequency is too low, the CPU might forget what it was doing. This is because the circuitry uses a number of dynamic nodes. A dynamic node is one that uses the note-to-ground capacitance to form a temporary storage cell. Dynamic nodes are used to reduce the transistor count, and hence the chip area, thus to produce a more economical device.

This is not to say that the on-chip RAM in CHMOS microcontrollers is dynamic. It's not. It's the CPU that is dynamic, and that is what imposes the minimum clock frequency specification.

LATCHUP

Latchup is an SCR-type turn-on phenomenon that is the traditional nemesis of CMOS systems. The substrate, the wells, and the transistors form parasitic pnpn structures within the device. These parasitic structures turn on like an SCR if a sufficient amount of forward current is driven through one of the junctions. From the circuit designer's point of view it can happen whenever an input or output pin is externally driven a diode drop above V_{CC} or below V_{SS}, by a source that is capable of supplying the required trigger current.

However much of a problem latchup has been in the past, it is good to know that in most recently developed CMOS devices, and specifically in CHMOS devices, the current required to trigger latchup is typically well over 100 mA. The 80C51BH is virtually immune to latchup. (References 1 and 2 present a discussion of the latchup mechanisms and the steps that are taken on the chip to guard against it.) Modern CMOS is not absolutely immune to latchup, but with trigger currents in the hundreds of mA, latchup is certainly a lot easier to avoid than it once was.

A careless power-up sequence might trigger a latchup in the older CMOS families, but it's unlikely to be a major problem in high-speed CMOS or in CHMOS. There is still some risk incurred in inserting or removing chips or boards in a CMOS system while the power is on. Also, severe transients, such as inductive kicks or momentary short-circuits, can exceed the trigger current for latchup.

For applications in which some latchup risk seems unavoidable, you can put a small resistor (100Ω or so) in series with signal lines to ensure that the trigger current will never be reached. This also helps to control overshoot and RFI.

LOGIC LEVELS AND INTERFACING PROBLEMS

CMOS logic levels differ from TTL levels in two ways. First, for equal supply voltages, CMOS gives (and requires) a higher "logic 1" level than TTL. Secondly, CMOS logic levels are V_{CC} (or VDD) dependent, whereas guaranteed TTL logic levels are fixed when V_{CC} is within TTL specs.

Standard 74HC logic levels are as follows:

$V_{IH}MIN$ = 70% of V_{CC}
$V_{IL}MAX$ = 20% of V_{CC}
$V_{OH}MIN = V_{CC} - 0.1V, |I_{OH}| \leq 20\ \mu A$
$V_{OL}MAX = 0.1V, |I_{OL}| \leq 20\ \mu A$

Figure 1 compares 74HC, LS TTL, and 74HCT logic levels with those of the HMOS 8051 and the CHMOS 80C51BH for V_{CC} = 5V.

Output logic levels depend of course on load current, and are normally specified at several load currents. When CMOS and TTL are powered by the same V_{CC}, the logic levels guaranteed on the data sheets indicate that CMOS can drive TTL, but TTL can't drive CMOS. The incompatibility is that the TTL circuit's V_{OH} level is too low to reliably be recognized by the CMOS circuit as a valid V_{IH}.

Since HMOS circuits were designed to be TTL-compatible, they have the same incompatibility.

Fortunately, 74HCT-series circuits are available to ease these interfacing problems. They have TTL-compatible logic levels at the inputs and standard CMOS levels at the outputs.

The 80C51BH is designed to work with either TTL or CMOS. Therefore its logic levels are specified very much like 74HCT circuits. That is, its input logic levels are TTL-compatible, and its output characteristics are like standard high-speed CMOS.

NOISE CONSIDERATIONS

One of the major reasons for going to CMOS has traditionally been that CMOS is less susceptible to noise. As previously noted, its low susceptibility to noise is

Logic State	V_{CC} = 5V				
	74HC	74HCT	LS TTL	8051	80C51BH
V_{IH}	3.5V	2.0V	2.0V	2.0V	1.9V
V_{IL}	1.0V	0.8V	0.8V	0.8V	0.9V
V_{OH}	4.9V	4.9V	2.7V	2.4V	4.5V
V_{OL}	0.1V	0.1V	0.5V	0.45V	0.45V

Figure 1. Logic Level Comparison. (Output voltage levels depend on load current. Data sheets list guaranteed output levels for several load currents. The output levels listed here are for minimum loading.)

partly due to superior noise margins, and partly due to its slow speed.

Noise margin is the difference between V_{OL} and V_{IL}, or between V_{OH} and V_{IH}. If V_{OH} from a driving circuit is 2.7V and V_{IH} to the driven circuit is 2.0V, then the driven circuit has 0.7V of noise margin at the logic high level. These kinds of comparisons show that an all-CMOS system has wider noise margins than an all-TTL system.

Figure 2 shows noise margins in CMOS and LS TTL systems when both have V_{CC} = 5V. It can be seen that CMOS/CMOS and CMOS/CHMOS systems have an edge over LS TTL in this respect.

Noise margins can be misleading, however, because they don't say how much noise energy it takes to induce in the circuit a noise voltage of sufficient amplitude to cause a logic error. This would involve consideration of the width of the noise pulse as compared with the circuit's response speed, and the impedance to ground from the point of noise introduction in the circuit.

When these considerations are included, it is seen that using the slower 74C- and 4000-series circuits with a 12 or 15V supply voltage does offer a truly improved level of noise immunity, but that high-speed CMOS at 5V is not significantly better than TTL.

One should not mistake the wider supply voltage tolerance of high-speed CMOS for V_{CC} glitch immunity. Supply voltage tolerance is a DC rating, not a glitch rating.

For any clocked CMOS, and most especially for VLSI CMOS, V_{CC} decoupling is critical. CHMOS draws current in extremely sharp spikes at the clock edges. The VHF and UHF components of these spikes are not drawn from the power supply, but from the decoupling capacitor. If the decoupling circuit is not sufficiently low in inductance, V_{CC} will glitch at each clock edge. We suggest that a 0.1 μF decoupler cap be used in a minimum-inductance configuration with the microcontroller. A minimum-inductance configuration is one that minimizes the area of the loop formed by the chip (V_{CC} to V_{SS}), the traces to the decoupler cap, and the decoupler cap. PCB designers too often fail to understand that if the traces that connect the decoupler cap to the V_{CC} and V_{SS} pins aren't short and direct, the decoupler loses much of its effectiveness.

Interface	Noise Margin for V_{CC} = 5V	
	Logic Low $V_{IL}-V_{OL}$	Logic High $V_{OH}-V_{IH}$
74HC to 74HC	0.9V	1.4V
LSTTL to LSTTL	0.3V	0.7V
LSTTL to 74HCT	0.3V	0.7V
LSTTL to 80C51BH	0.3V	0.7V
74HC to 80C51BH	0.8V	3.0V
80C51BH to 74HC	0.8V	1.0V

Figure 2. Noise Margins for CMOS and LS TTL Circuits

Overshoot and ringing in signal lines are potential sources of logic upsets. These can largely be controlled by circuit layout. Inserting small resistors (about 100Ω) in series with signal lines that seem to need them will also help.

The sharp edges produced by high-speed CMOS can cause RFI problems. The severity of these problems is largely a function of the PCB layout. We don't mean to imply that all RFI problems can be solved by a better PCB layout. It may well be, for example, that in some RFI-sensitive designs high-speed CMOS is simply not the answer. But circuit layout is a critical factor in the noise performance of any electronic system, and more so in high-speed CMOS systems than others.

Circuit layout techniques for minimizing noise susceptibility and generation are discussed in References 3 through 6.

UNUSED PINS

CMOS input pins should not be left to float, but should always be pulled to one logic level or the other. If they float, they tend to float into the transition region between 0 and 1, where the pullup and pulldown devices in the input buffer are both conductive. This causes a significant increase in I_{CC}. A similar effect exists in HMOS circuits, but with less noticeable results.

In 80C51BH and 80C31BH designs, unused pins of Ports 1, 2, and 3 can be ignored, because they have internal pullups that will hold them at a valid Logic 1 level. Port 0 pins are different, however, in not having internal pullups (except during bus operations).

When the 80C51BH is in reset, the Port 0 pins are in a float state unless they are externally pulled up or down. If it's going to be held in reset for just a short time, the transient float state can probably be ignored. When it comes out of reset, the pins stay afloat unless

they are externally pulled either up or down. Alternatively, the software can internally write 0s to whatever Port 0 pins may be unused.

The same considerations are applicable to the 80C31BH with regards to reset. But when the 80C31BH comes out of reset, it commences bus operations, during which the logic levels at the pins are always well defined as high or low.

Consider the 80C31BH in the Power Down or Idle modes, however. In those modes it is not fetching instructions, and the Port 0 pins will float if not externally pulled high or low. The choice of whether to pull them high or low is the designer's. Normally it is sufficient to pull them up to V_{CC} with 10k resistors. But if power is going to be removed from circuits that are connected to the bus, it will be advisable to pull the bus pins down (normally with 10k resistors). Considerations involved in selecting pullup and pulldown resistor values are as follows.

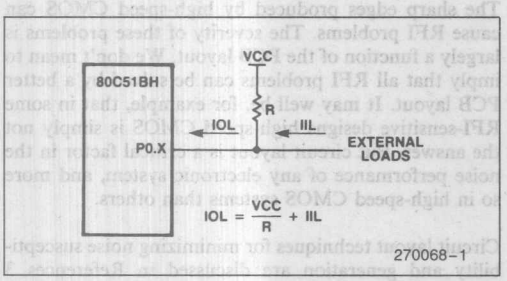


Figure 3a. Conditions defining the minimum value for R. P0.X is emitting a logic low. R must be large enough to not cause IOL to exceed data sheet specifications.

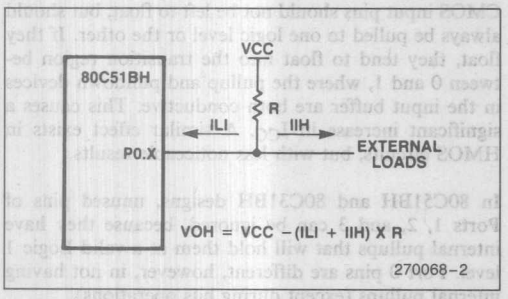


Figure 3b. Conditions defining the maximum value for R. P0.X is in a high impedance state. R must be small enough to keep V_{OH} acceptably high.

PULLUP RESISTORS

If a pullup resistor is to be used on a Port 0 pin, its minimum value is determined by I_{OL} requirements. If the pin is trying to emit a 0, then it will have to sink the current from the pullup resistor plus whatever other current may be sourced by other loads connected to the pin, as shown in Figure 3a, while maintaining a valid output low (V_{OL}). To guarantee that the pin voltage will not exceed 0.45V, the resistor should be selected so that I_{OL} doesn't exceed the value specified on the data sheet. In most CMOS applications, the minimum value would be about 2k Ω.

The maximum value you could use depends on how fast you want the pin to pull up after bus operations have ceased, and how high you want the V_{OH} level to be. The smaller the resistor the faster it pulls up. Its effect on the V_{OH} level is that $V_{OH} = V_{CC} - (ILI + IIH) \times R$. ILI is the input leakage current to the Port 0 pin, and IIH is the input high current to the external loads, as shown in Figure 3b. Normally V_{OH} can be expected to reach 0.9 V_{CC} if the pullup resistance does not exceed about 50k Ω.

Pulldown Resistors

If a pulldown resistor is to be used on a Port 0 pin, its minimum value is determined by V_{OH} requirements during bus operations, and its maximum value is in most cases determined by leakage current.

During bus operations the port uses internal pullups to emit 1s. The D.C. Characteristics in the data sheet list guaranteed V_{OH} levels for given I_{OH} currents. (The "-" sign in the I_{OH} value means the pin is sourcing that current to the external load, as shown in Figure 4.) To ensure the V_{OH} level listed in the data sheet, the resis-

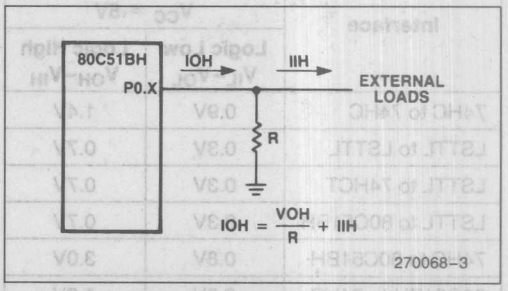


Figure 4a. Conditions defining the minimum value for R. P0.X is emitting a 1 in a bus operation. R must be large enough to not cause IOH to exceed data sheet specifications.

$$\frac{V_{OH}}{R} + I_{IH} \leq |I_{OH}|$$

tor has to satisfy where I_{IH} is the input high current to the external loads.

When the pin goes into a high impedance state, the pulldown resistor will have to sink leakage current from the pin, plus whatever other current may be sourced by other loads connected to the pin, as shown in Figure 4b. The Port 0 leakage current is I_{LI} on the data sheet. The resistor should be selected so that the voltage developed across it by these currents will be seen as a logic low by whatever circuits are connected to it (including the 80C51BH). In CMOS/CHMOS applications, 50k Ω is normally a reasonable maximum value.

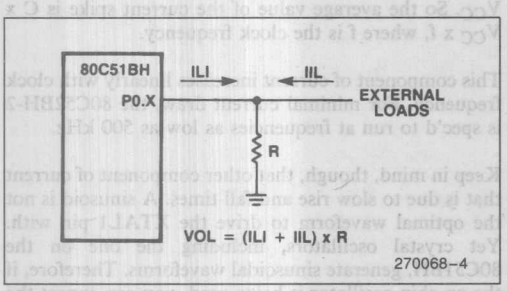


Figure 4b. Conditions defining the maximum value for R. P0.X is in a high impedance state. R must be small enough to keep VOL acceptably low.

DRIVE CAPABILITY OF THE INTERNAL PULLUPS

There's an important difference between HMOS and CHMOS port drivers. The pins of Ports 1, 2, and 3 of the CHMOS parts each have three pullups: strong, normal, and weak, as shown in Figure 5. The strong pullup (p1) is only used during 0-to-1 transitions, to hasten the transition. The weak pullup (p2) is on whenever the bit latch contains a 1. The "normal" pullup (p3) is controlled by the pin voltage itself.

The reason that p3 is controlled by the pin voltage is that if the pin is being used as an input, and the external source pulls it to a low, then turning off p3 makes for a lower I_{IL}. The data sheet shows an "I_{TL}" specification. This is the current that p3 will source during the time the pin voltage is making its 1-to-0 transition. This is what I_{IL} would be if an input low at the pin didn't turn p3 off.

Note, however, that this p3 turn-off mechanism puts a restriction on the drive capacity of the pin if it's being used as an output. If you're trying to output a logic high, and the external load pulls the pin voltage below the pin's V_{IH}MIN spec, p3 might turn off, leaving only the weak p2 to provide drive to the load. To prevent this happening, you need to ensure that the load doesn't draw more than the I_{OH} spec for a valid V_{OH}. The idea is to make sure the pin voltage never falls below its own V_{IH}MIN specification.

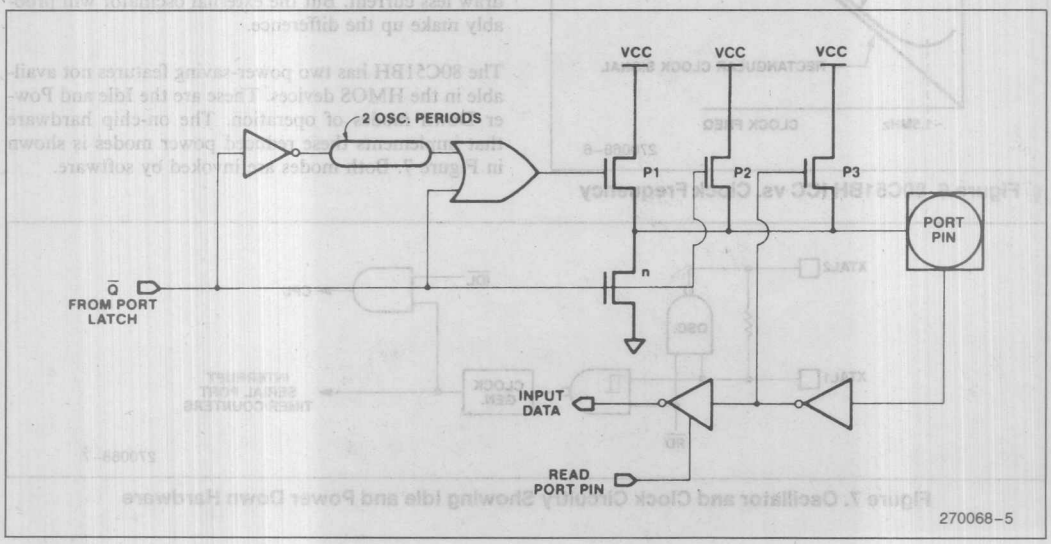


Figure 5. 80C51BH Output Drivers for Ports 1, 2 and 3

POWER CONSUMPTION

The main reason for going to CMOS, of course, is to conserve power. (There are other reasons, but this is the main one.) Conserving power doesn't mean just reducing your electric bill. Nor does it necessarily relate to battery operation, although battery operation without CMOS is pretty unhandy. The main reason for conserving power is to be able to put more functionality into a smaller space. The reduced power consumption allows the use of smaller and lighter power supplies, and less heat being generated allows denser packaging of circuit components. Expensive fans and blowers can usually be eliminated.

A cooler running chip is also more reliable, since most random and wearout failures relate to die temperature. And finally, the lower power dissipation will allow more functions to be integrated onto the chip.

The reason CMOS consumes less power than NMOS is that when it's in a stable state there is no path of conduction from V_{CC} to V_{SS} except through various leakage paths. CMOS does draw current when it's changing states. How much current it draws depends on how often and how quickly it changes states.

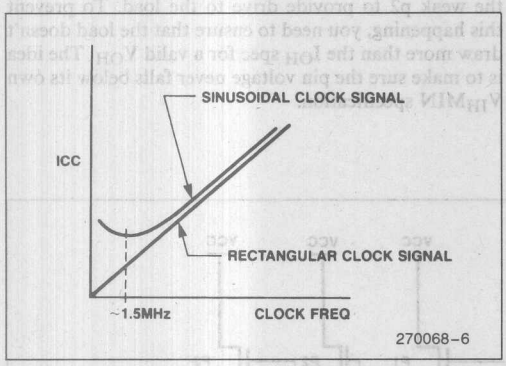


Figure 6. 80C51BH ICC vs. Clock Frequency

CMOS circuits draw current in sharp spikes during logical transitions. These current spikes are made up of two components. One is the current that flows during the transition time when pullup and pulldown FETs are both active. The average (DC) value of this component is larger when the transition times of the input signals are longer. For this reason, if the current draw is a critical factor in the design, slow rise and fall times should be avoided, even when the system speed doesn't seem to justify a need for nanosecond switching speeds.

The other component is the current that charges stray and load capacitance at the nodes of a CMOS logic gate. The average value of this current spike is its area (integral over time) multiplied by its rep rate. Its area is the amount of charge it takes to raise the node capacitance, C, to V_{CC}. That amount of charge is just C x V_{CC}. So the average value of the current spike is C x V_{CC} x f, where f is the clock frequency.

This component of current increases linearly with clock frequency. For minimal current draw, the 80C52BH-2 is spec'd to run at frequencies as low as 500 kHz.

Keep in mind, though, that other component of current that is due to slow rise and fall times. A sinusoid is not the optimal waveform to drive the XTAL1 pin with. Yet crystal oscillators, including the one on the 80C51BH, generate sinusoidal waveforms. Therefore, if the on-chip oscillator is being used, you can expect the device to draw more current at 500 kHz, than it does at 1.5 MHz, as shown in Figure 6. If you derive a good sharp square wave from an external oscillator, and use that to drive XTAL1, then the microcontroller will draw less current. But the external oscillator will probably make up the difference.

The 80C51BH has two power-saving features not available in the HMOS devices. These are the Idle and Power Down modes of operation. The on-chip hardware that implements these reduced power modes is shown in Figure 7. Both modes are invoked by software.

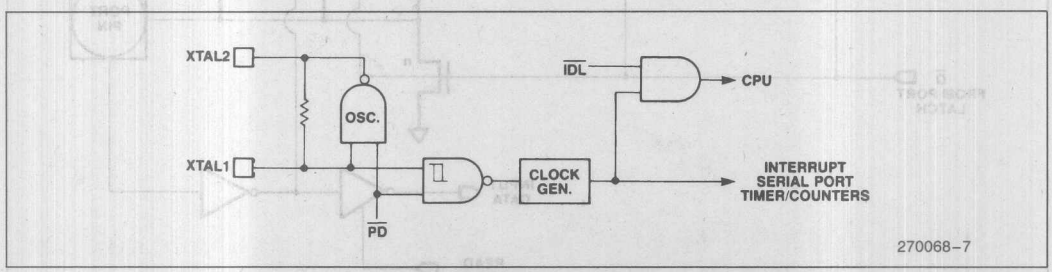


Figure 7. Oscillator and Clock Circuitry Showing Idle and Power Down Hardware

Idle: In the Idle Mode ($\overline{IDL}$ = 0 in Figure 7), the CPU puts itself to sleep by gating off its own clock. It doesn't stop the oscillator. It just stops the internal clock signal from getting to the CPU. Since the CPU draws 80 to 90 percent of the chip's power, shutting it off represents a fairly significant power savings. The on-chip periperals (timers, serial port, interrupts, etc.) and RAM continue to function as normal. The CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle.

The Idle Mode is invoked by setting bit 0 (IDL) of the PCON register. PCON is not bit-addressable, so the bit has to be set by a byte operation, such as

```
ORL PCON,#1
```

The PCON register also contains flag bits GF0 and GF1, which can be used for any general purposes, or to give an indication if an interrupt occurred during normal operation or during Idle. In this application, the instruction that invokes Idle also sets one or both of the flag bits. Their status can then be checked in the interrupt routines.

While the device is in the Idle Mode, ALE and $\overline{PSEN}$ emit logic high (V_{OH}), as shown in Figure 8. This is so external EPROM can be deselected and have its output disabled.

The port pins hold the logical states they had at the time the Idle was activated. If the device was executing out of external program memory, Port 0 is left in a high impedance state and Port 2 continues to emit the high byte of the program counter (using the strong pullups to emit 1s). If the device was executing out of internal program memory, Ports 0 and 2 continue to emit whatever is in the P0 and P2 registers.

There are two ways to terminate Idle. Activation of any enabled interrupt will cause the hardware to clear bit 0 of the PCON register, terminating the Idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that invoked Idle.

The other way is with a hardware reset. Since the clock oscillator is still running, RST only needs to be held active for two machine cycles (24 oscillator periods) to complete the reset. Note that this exit from Idle writes 1s to all the ports, initializes all SFRs to their reset values, and restarts program execution from location 0.

Power Down: In the Power Down Mode ($\overline{PD}$ = 0 in Figure 7), the CPU puts the whole chip to sleep by turning off the oscillator. In case it was running from an external oscillator, it also gates off the path to the internal phase generators, so no internal clock is generated even if the external oscillator is still running. The on-chip RAM, however, saves its data, as long as V_{CC} is maintained. In this mode the only I_{CC} that flows is leakage, which is normally in the micro-amp range.

The Power Down Mode is invoked by setting bit 1 in the PCON register, using a byte instruction such as

```
ORL   PCON,#2
```

While the device is in Power Down, ALE and $\overline{PSEN}$ emit lows (V_{OL}), as shown in Figure 8. The reason they are designed to emit lows is so that power can be removed from the rest of the circuit, if desired, while the 80CS51BH is in its Power Down mode.

The port pins continue to emit whatever data was written to them. Note that Port 2 emits its P2 register data even if execution was from external program memory.

Pin	Internal Execution		External Execution	
	Idle	Power Down	Idle	Power Down
ALE	1	0	1	0
PSEN/	1	0	1	0
P0	SFR Data	SFR Data	High-Z	High-Z
P1	SFR Data	SFR Data	SFR Data	SFR Data
P2	SFR Data	SFR Data	PCH	SFR Data
P3	SFR Data	SFR Data	SFR Data	SFR Data

Figure 8. Status of Pins in Idle and Power Down Modes. "SFR data" means the port pins emit their internal register data. "PCH" is the high byte of the Program Counter.

Port 0 also emits its P0 register data, but if execution was from external program memory, the P0 register data is FF. The oscillator is stopped, and the part remains in this state as long as V_{CC} is held, and until it receives an external reset signal.

The only exit from Power Down is a hardware reset. Since the oscillator was stopped, RST must be held active long enough for the oscillator to re-start and stabilize. Then the reset function initializes all the Special Function Registers (ports, timers, etc.) to their reset values, and re-starts the program from location 0. Therefore, timer reloads, interrupt enables, baud rates, port status, etc. need to be re-established. Reset does not affect the content of the on-chip data RAM. If V_{CC} was held during Power Down, the RAM data is still good.

USING THE POWER DOWN MODE

The software-invoked Power Down feature offers a means of reducing the power consumption to a mere trickle in systems which are to remain dormant for some period of time, while retaining important data.

The user should give some thought to what state the port pins should be left in during the time the clock is stopped, and write those values to the port latches before invoking Power Down.

If V_{CC} is going to be held to the entire circuit, one would want to write values to the port latches that would deselect peripherals before invoking Power Down. For example, if external memory is being used, the P2 SFR should be loaded with a value which will not generate an active chip select to any memory device.

In some applications, V_{CC} to part of the system may be shut off during Power Down, so that even quiescent and standby currents are eliminated. Signal lines that connect to those chips must be brought to a logic low, whether the chip in question is CMOS, NMOS, or TTL, before V_{CC} is shut off to them. CMOS pins have parasitic pn junctions to V_{CC}, which will be forward biased if V_{CC} is reduced to zero while the pin is held at a logic high. NMOS pins often have FETs that look like diodes to V_{CC}. TTL circuits may actually be damaged by an input high if $V_{CC} = 0$. That's why the 80C51BH outputs lows at ALE and $\overline{PSEN}$ during Power Down.

Figure 9 shows a circuit that can be used to turn V_{CC} off to part of the system during Power Down. The circuit will ensure that the secondary circuit is not de-energized until after the 80C31BH is in Power Down, and that the 80C31BH does not receive a reset (terminating the Power Down mode) before the secondary circuit is re-energized. Therefore, the program memory itself can be part of the secondary circuit.

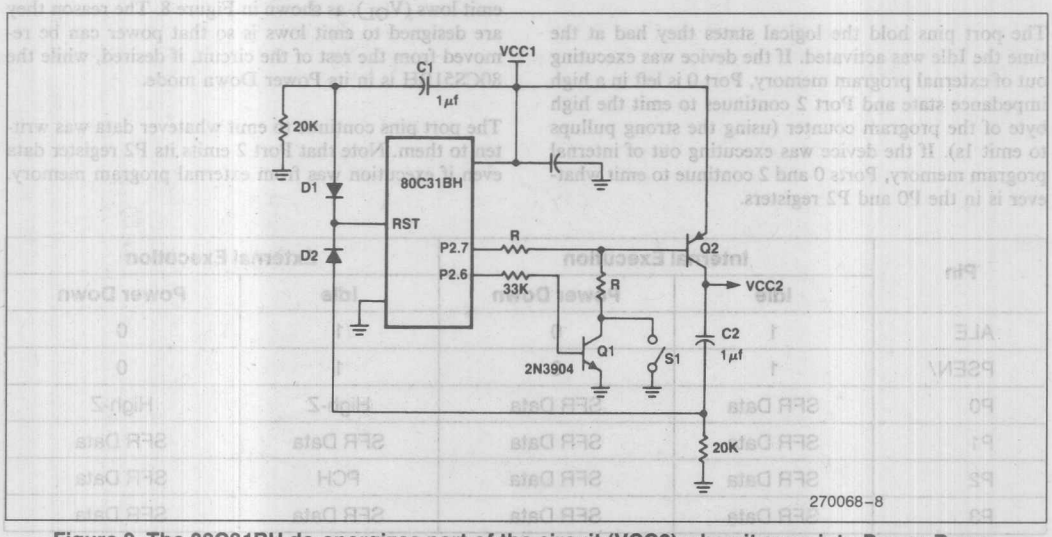


Figure 9. The 80C31BH de-energizes part of the circuit (VCC2) when it goes into Power Down. Selections of R and Q2 depend on VCC2 current draw.

In Figure 9, when V_{CC} is switched on to the 80C31BH, capacitor C1 provides a power-on reset. The reset function writes 1s to all the port pins. The 1 at P2.6 turns Q1 on, enabling V_{CC} to the secondary circuit through transistor Q2. As the 80C31BH comes out of reset, Port 2 commences emitting the high byte of the Program Counter, which results in the P2.7 and P2.6 pins outputting 0s. The 0 at P2.7 ensures continuation of V_{CC} to the secondary circuit.

The system software must now write a 1 to P2.7 and a 0 to P2.6 in the Port 2 SFR, P2. These values will not appear at the Port 2 pins as long as the device is fetching instructions from external program memory. However, whenever the 80C31BH goes into Power Down, these values will appear at the port pins, and will shut off both transistors, disabling V_{CC} to the secondary circuit.

Closing the switch S1 re-energizes the secondary circuit, and at the same time sends a reset through C2 to the 80C31BH to wake it up. The diode D1 is to prevent C1 from hogging current from C2 during this secondary reset. D2 prevents C2 from discharging through the RST pin when V_{CC} to the secondary circuit goes to zero.

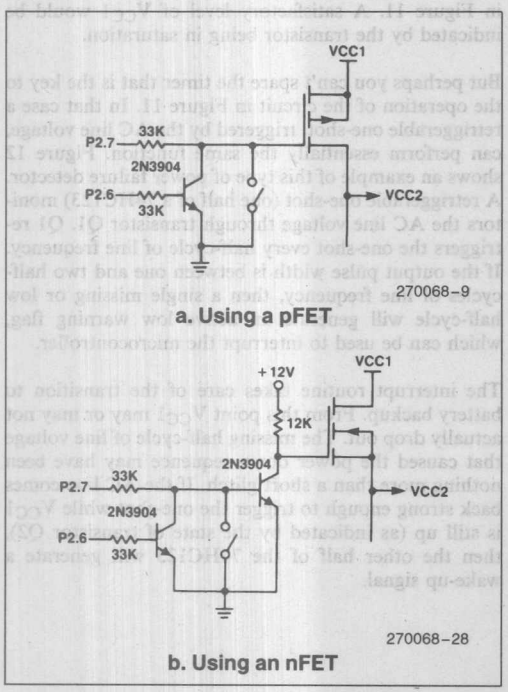


Figure 10. Using Power MOSFETs to Control VCC2

USING POWER MOSFETs to CONTROL V_{CC}

Power MOSFETs are gaining in popularity (and availability). The easiest way to control V_{CC} is with a Logic Level pFET, as shown in Figure 10a. This circuit allows the full V_{CC} to be used to turn the device on. Unfortunately, power pFETs are not economically competitive with bipolar transistors of comparable ratings.

Power nFETs are both economical and available, and can be used in this application if a DC supply of higher voltage is available to drive the gate. Figure 10b shows how to implement a V_{CC} switch using a power nFET and a (nominally) + 12V supply. The problem here is that if the device is on, its source voltage is +5V. To maintain the on state, the gate has to be another 5 or 10V above that. The "12V" supply is not particularly critical. A minimally filtered, unregulated rectifier will suffice.

BATTERY BACKUP SYSTEMS

Here we consider circuits that normally draw power from the AC line, but switch to battery operation in the event of a power failure. We assume that in battery operation high-current loads will be allowed to die along with the AC power. The system may continue then with reduced functionality, monitoring a control transducer, perhaps, or driving an LCD. Or it may go into a bare-bones survival mode, in which critical data is saved but nothing else happens till AC power is restored.

In any case it is necessary to have some early warning of an impending power failure so that the system can arrange an orderly transfer to battery power. Early warning systems can operate by monitoring either the AC line voltage or the unregulated rectifier output, or even by monitoring the regulated DC voltage.

Monitoring the AC line voltage gives the earliest warning. That way you can know within one or two half-cycles of line frequency that AC power is down. In most cases you then have at least another half-cycle of line frequency before the regulated V_{CC} starts to fall. In a half-cycle of line frequency an 80C51BH can execute about 5,000 instructions—plenty of time to arrange an orderly transfer of power.

The circuit in Figure 11 uses a Zener diode to test the line voltage each half-cycle, and a junction transistor to pass the information on to the 80C51BH. (Obviously a voltage comparator with a suitable reference source can

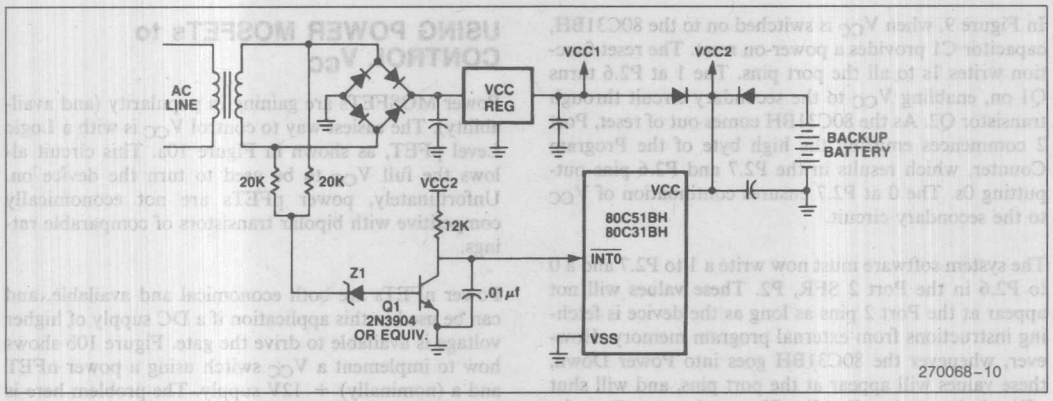


Figure 11. Power Failure Detector with Battery Backup. When AC power fails, VCC1 goes down and VCC2 is held.

perform the same function, if one prefers.) The way it works is if the line voltage reaches an acceptably high level, it breaks over Z1, drives Q1 to saturation, and interrupts the 80C51BH. The interrupt would be transition-activated, in this application. The interrupt service routine reloads one of the C51BH's timers to a value that will make it roll over in something between one and two half-cycles of line frequency. As long as the line voltage is healthy, the timer never rolls over, because it is reloaded every half-cycle. If there is a single half-cycle in which the line voltage doesn't reach a high enough level to generate the interrupt, the timer rolls over and generates a timer interrupt.

The timer interrupt then commences the transition to battery backup. Critical data needs to be copied into protected RAM. Signals to circuits that are going to lose power must be written to logic low. Protected circuits (those powered by $V_{CC}2$) that communicate with unprotected circuits must be deselected. The microcontroller itself may be put into Idle, so that it can continue some level of interrupt-driven functionality, or it may be put into Power Down.

Note that if the CPU is going to invoke Power Down, the Special Function Registers may also need to be copied into protected RAM, since the reset that terminates the Power Down mode will also intialize all the SFRs to their reset values.

The circuit in Figure 11 does not show a wake-up mechanism. A number of choices are available, however. A pushbutton could be used to generate an interrupt, if the CPU is in Idle, or to activate reset, if the CPU is in Power Down.

Automatic wake-up on power restoration is also possible. If the CPU is in Idle, it can continue to respond to any interrupts that might be generated by Q1. The interrupt service routine determines from the status of flag bits GF0 and GF1 in PCON that it is in Idle because there was a power outage. It can then sample $V_{CC}1$ through a voltage comparator similar to Z1, Q1 in Figure 11. A satisfactory level of $V_{CC}1$ would be indicated by the transistor being in saturation.

But perhaps you can't spare the timer that is the key to the operation of the circuit in Figure 11. In that case a retriggerable one-shot, triggered by the AC line voltage, can perform essentially the same function. Figure 12 shows an example of this type of power failure detector. A retriggerable one-shot (one half of a 74HC123) monitors the AC line voltage through transistor Q1. Q1 retriggers the one-shot every half-cycle of line frequency. If the output pulse width is between one and two half-cycles of line frequency, then a single missing or low half-cycle will generate an active low warning flag, which can be used to interrupt the microcontroller.

The interrupt routine takes care of the transition to battery backup. From this point $V_{CC}1$ may or may not actually drop out. The missing half-cycle of line voltage that caused the power down sequence may have been nothing more than a short glitch. If the AC line comes back strong enough to trigger the one-shot while $V_{CC}1$ is still up (as indicated by the state of transistor Q2), then the other half of the 74HC123 will generate a wake-up signal.

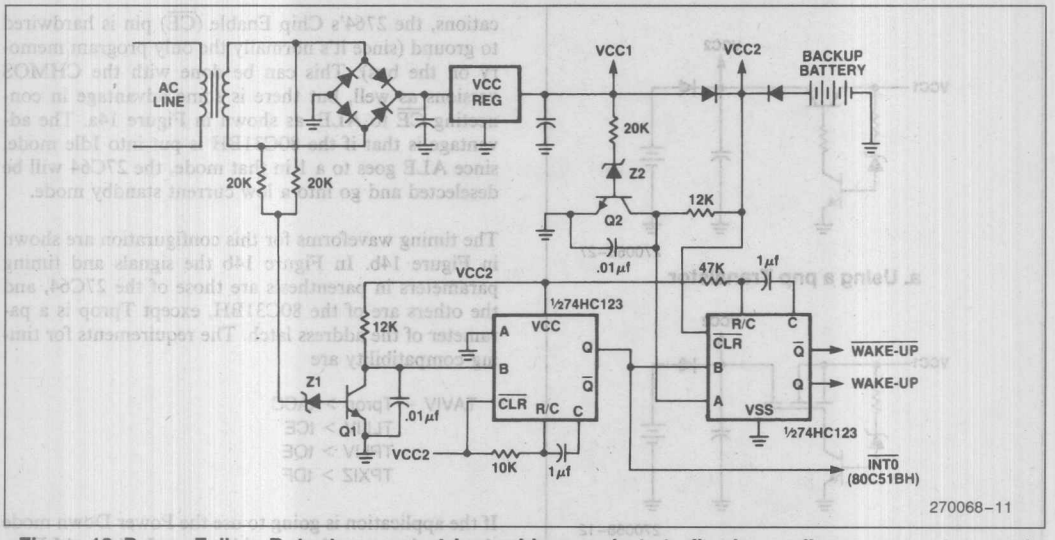


Figure 12. Power Failure Detector uses retriggerable one-shots to flag impending power outage and generate automatic wake-up when power returns.

Having been awakened, the 80C51BH will stay awake for at least another half-cycle of line frequency (another 5,000 or so instructions) before possibly being told to arrange another transfer of power. Consequently, if the line voltage is jittering erratically around the switch-over point (determined by diode Z1), the system will limp along executing in half-cycle units of line frequency.

On the other hand, if the power outage is real and lengthy, $V_{CC}1$ will eventually fall below the level at which the backup battery takes over. The backup battery maintains power to the 80C51BH, and to the 74HC123, and to whatever other circuits are being protected during this outage. The battery voltage must be high enough to maintain $V_{CC}MIN$ specs to the 80C51BH.

If the microcontroller is an 80C31BH, executing out of external ROM, and if the C31BH is put into Idle during the power outage, then the external ROM must also be supplied by the battery. On the other hand, if the C31BH is put into Power Down during the outage, then the ROM can be allowed to die with the AC power. The considerations here are the same as in Figure 9: V_{CC} to the ROM is still up at the time Power Down is invoked, and we must ensure (through selection of diode Z2 in Figure 12) that the 80C31BH is not awakened till ROM power is back in spec.

POWER SWITCHOVER CIRCUITS

Battery backup systems need to have a way for the protected circuits to draw power from the line-operated power supply when that source is available, and to switch over to battery power when required. The switchover circuit is simple if the entire system is to be battery powered in the event of a line power outage. In that case a pair of diodes suffice, as shown in Figure 12, provided $V_{CC}MIN$ specs are still met after the diode drop has been subtracted from its respective power source.

The situation becomes more complicated when part of the circuit is going to be allowed to die when the AC power goes out. In that case it is difficult to maintain equal V_{CC}s to protected and unprotected circuits (and possibly dangerous not to).

The problem can be alleviated by using a Schottky diode instead of a 1N4001, for its lower forward voltage drop. The 1N5820, for example, has a foward drop of about 0.35V at 1A.

Other solutions are to use a transistor or power MOSFET switch, as shown in Figure 13. With minor modifications this switch can be controlled by port pins.

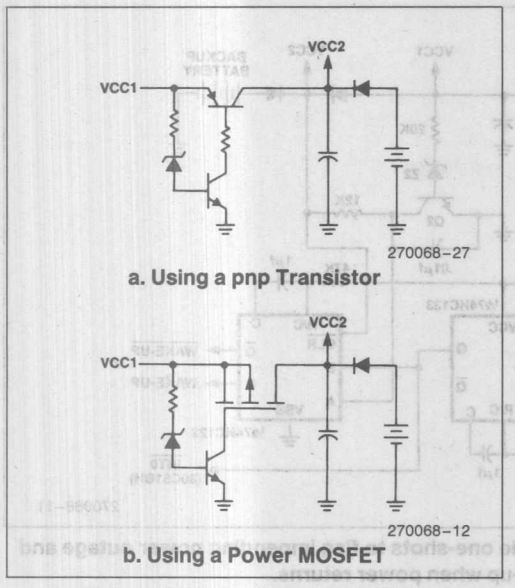


Figure 13. Power Switchover Ckts.

80C31BH + CHMOS EPROM

The 27C64 and 87C64 are Intel's 8K byte CHMOS EPROMs. The 27C64 requires an external address latch, and can be used with the 80C31BH as shown in Figure 14a. In most 8031 + 2764 (HMOS) applications, the 2764's Chip Enable ($\overline{CE}$) pin is hardwired to ground (since it's normally the only program memory on the bus). This can be done with the CHMOS versions as well, but there is some advantage in connecting $\overline{CE}$ to ALE, as shown in Figure 14a. The advantage is that if the 80C31BH is put into Idle mode, since ALE goes to a 1 in that mode, the 27C64 will be deselected and go into a low current standby mode.

The timing waveforms for this configuration are shown in Figure 14b. In Figure 14b the signals and timing parameters in parenthesis are those of the 27C64, and the others are of the 80C31BH, except Tprop is a parameter of the address latch. The requirements for timing compatibility are

TAVIV − Tprop > tACC
TLLIV > tCE
TPLIV > tOE
TPXIZ > tDF

If the application is going to use the Power Down mode then we have another consideration: In Idle, ALE = $\overline{PSEN}$ = 1, and in Power Down, ALE = $\overline{PSEN}$ = 0. In a realistic application there are likely to be more chips in the circuit than are shown in Figure 14, and it is likely that the nonessential ones will have their V_{CC} removed while the CPU is in Power Down. In that case the EPROM and the address latch should be among the chips that have V_{CC} removed, and logic lows are exactly what are required at ALE and $\overline{PSEN}$.

But if V_{CC} is going to be maintained to the EPROM during Power Down, then it will be necessary to de-

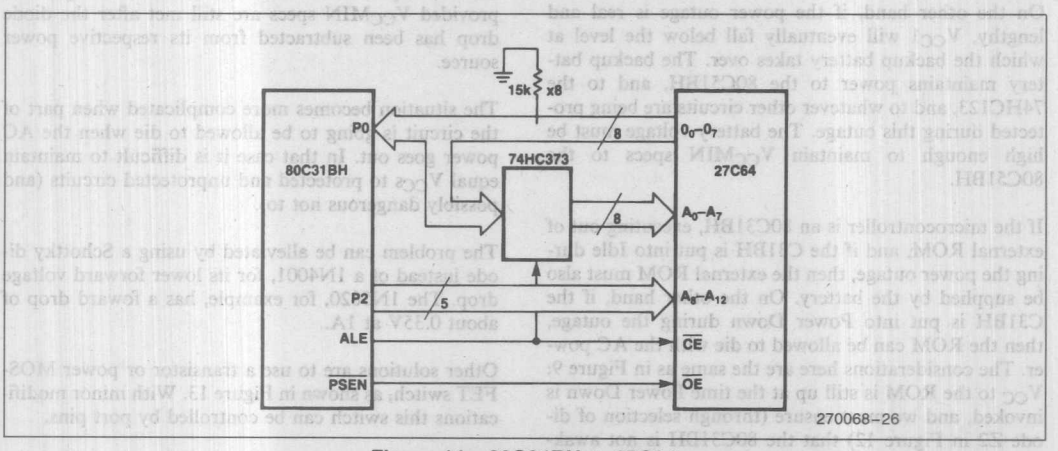


Figure 14a. 80C31BH + 27C64

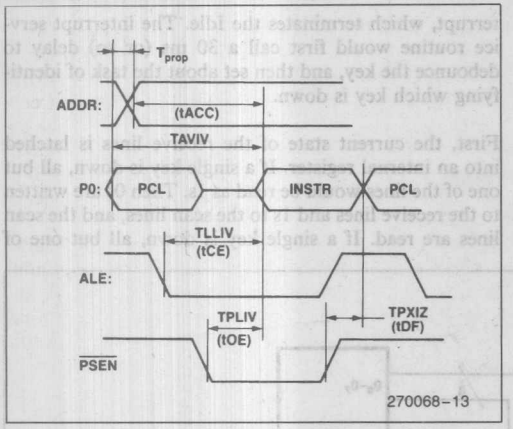


Figure 14b. Timing Waveforms for 80C31BH + 27C64

select the EPROM when the CPU is in Power Down. If Idle is never invoked, $\overline{CE}$ of the EPROM can be connected to P2.7 of the 80C31BH, as shown in Figure 15a. In normal operation, P2.7 will be emitting the MSB of the Program Counter, which is 0 if the program contains less than 32K of code. Then when the CPU goes into Power Down, the Port 2 pins emit P2 SFR data, which puts a 1 at P2.7, thus deselecting the EPROM.

If Idle and Power Down are both going to be used, $\overline{CE}$ of the EPROM can be driven by the logical OR of ALE and P2.7, as shown in Figure 15b. In Idle, ALE = 1 will deselect the EPROM, and in Power Down, P2.7 = 1 will deselect it.

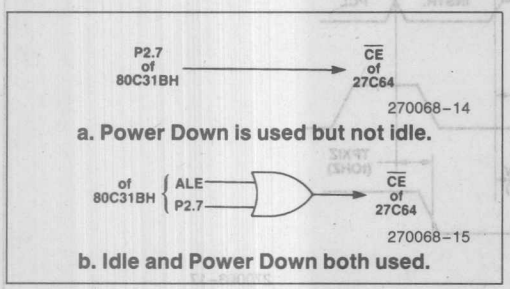


Figure 15. Modifications to 80C31BH/27C64 Interface

Pulldown resistors are shown in Figure 14a under the assumption that something on the bus is going to have its V_{CC} removed during Power Down. If this is not the case, pullups can be used as well as pulldowns.

The 87C64 is like the 27C64 except that it has an on-chip address latch. The Port 0 pins are tied to both address and data pins of the 87C64, as shown in Figure 16a. ALE drives the EPROM's ALE/$\overline{CS}$ input. During ALE high, the address information is allowed to flow into the EPROM and begin accessing the code byte. On the falling edge of ALE the address byte is internally latched. The A0–A7 inputs are then ignored and the same bus lines are used to transmit the fetched code byte from the O0–O7 pins back to the 80C31BH.

The timing waveforms for this configuration are shown in Figure 16b. In Figure 16b the signals and timing parameters in parentheses are those of the 87C64, and the others are of the 80C31BH. The requirements for timing compatibility are

TLHLL > tLL

TAVLL > tAL

TLLAX > tLA

TLLIV > tACL

TPLIV > tOE

TLLPL > tCOE

TPXIZ > tOHZ

The same considerations apply to the 87C64 as to the 27C64 with regards to the Idle and Power Down modes. Basically you want $\overline{CS}$ = 1 if V_{CC} is maintained to the EPROM, and $\overline{CS}$ = $\overline{OE}$ = 0 if V_{CC} is removed.

SCANNING A KEYBOARD

There are many different kinds of keyboards, but alphanumeric keyboards generally consist of a matrix of 8 scan lines and 8 receive lines as shown in Figure 17. Each set of lines connects to one port of the microcontroller. The software has written 0s to the scan lines, and 1s to the receive lines. Pressing a key connects a scan line to a receive line, thus pulling the receive line to a logic low.

The 8 receive lines are ANDed to one of the external interrupt pins, so that pulling any of the receive lines low generates an interrupt. The interrupt service routine has to identify the pressed key, if only one key is down, and convert that information to some useful output. If more than one key in the line matrix is found to be pressed, no action is taken. (This is a "two key lockout" scheme.)

cape, etc.) are not a part of the line matrix. These keys would connect directly to a port pin on the microcontroller, and would not cause lock-out if pressed simultaneously with a matrix key, nor generate an interrupt if pressed singly.

Normally the microcontroller would be in idle mode when a key has not been pressed, and another task is not in progress. Pressing a matrix key generates an interrupt, which terminates the idle. The interrupt service routine would first call a 30 ms (or so) delay to debounce the key, and then set about the task of identifying which key is down.

First, the current state of the receive lines is latched into an internal register. If a single key is down, all but one of the lines would be read as 1s. Then 0s are written to the receive lines and 1s to the scan lines, and the scan lines are read. If a single key is down, all but one of

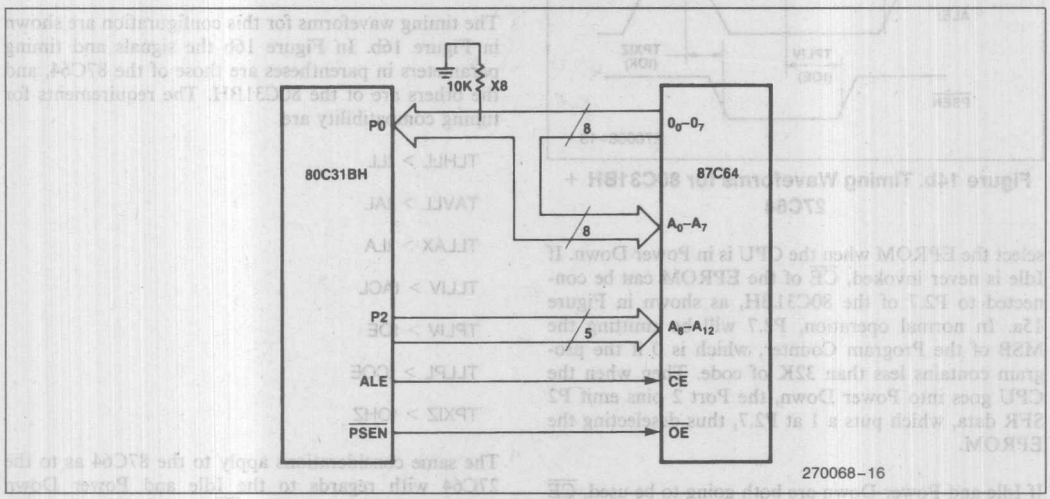


Figure 16a. 80C31BH + 87C64

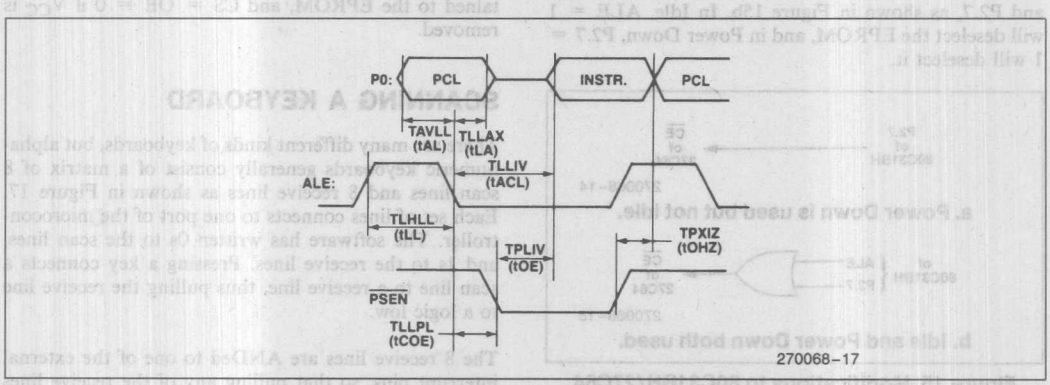


Figure 16b. Timing Waveforms for 80C31BH + 87C64

these lines would be read as 1s. By locating the single 0 in each set of lines, the pressed key can be identified. If more than one matrix key is down, one or both sets of lines will contain multiple 0s.

A subroutine is used to determine which of 8 bits in either set of lines is 0, and whether more than one bit is 0. Figure 18 shows a subroutine (SCAN) which does that using the 8051's bit-addressing capability. To use the subroutine, move the line data into a bit-addressable RAM location named LINE, and call the SCAN routine. The number of LINE bits which are zero is returned in ZERO__COUNTER. If only one bit is zero, its number (1 through 8) is returned in ZERO__BIT.

The interrupt service routine that is executed in response to a key closure might then be as follows:

```
RESPONSE_TO_KEY_CLOSURE:
        CALL    DEBOUNCE_DELAY
        MOV     LINE,P1; ;See Figure 17.
        CALL    SCAN
        DJNZ    ZERO_COUNTER,REJECT
        MOV     ADDRESS,ZERO_BIT
        MOV     P2,#0FFH; ;See Figure 17.
        MOV     P1,#0
        MOV     LINE,P2
        CALL    SCAN
        DJNZ    ZERO_COUNTER,REJECT
        XCH     A,ZERO_BIT
        SWAP    A
        ORL     ADDRESS,A
        XCH     A,ZERO_BIT
        MOV     P1,#0FFH
        MOV     P2,#0
REJECT: CLR     EX0
        RETI
```

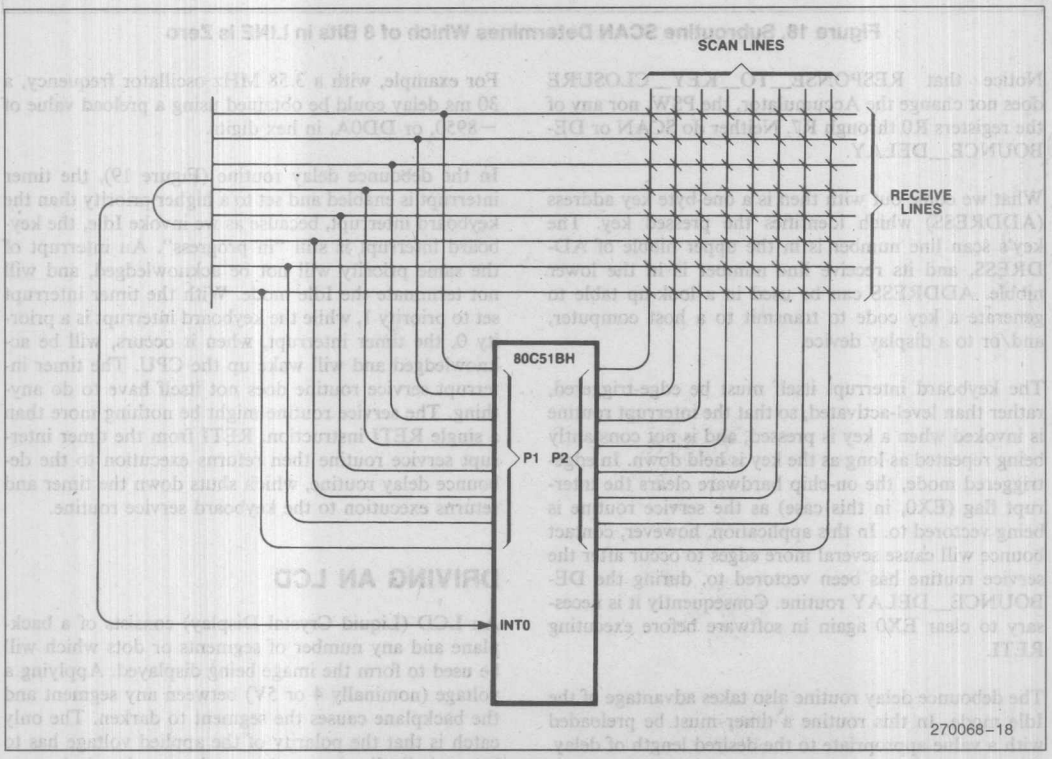


Figure 17. Scanning a Keyboard

```
SCAN:   MOV     ZERO_COUNTER,#0 ; ZERO_COUNTER counts the number of 0s in LINE.
        JB      LINE.0,ONE      ; Test LINE bit 0.
        INC     ZERO_COUNTER    ; If LINE.0 = 0, increment ZERO_COUNTER
        MOV     ZERO_BIT,#1     ;    and record that line number 1 is active.
ONE:    JB      LINE.1,TWO      ; Procedure continues for other LINE bits.
        INC     ZERO_COUNTER
        MOV     ZERO_BIT,#2     ; Line number 2 is active.
TWO:    JB      LINE.2,THREE
        INC     ZERO_COUNTER
        MOV     ZERO_BIT,#3     ; Line number 3 is active.
THREE:  JB      LINE.3,FOUR
        INC     ZERO_COUNTER
        MOV     ZERO_BIT,#4     ; Line number 4 is active.
FOUR:   JB      LINE.4,FIVE
        INC     ZERO_COUNTER
        MOV     ZERO_BIT,#5     ; Line number 5 is active.
FIVE:   JB      LINE.5,SIX
        INC     ZERO_COUNTER
        MOV     ZERO_BIT,#6     ; Line number 6 is active.
SIX:    JB      LINE.6,SEVEN
        INC     ZERO_COUNTER
        MOV     ZERO_BIT,#7     ; Line number 7 is active.
SEVEN:  JB      LINE.7,EIGHT
        INC     ZERO_COUNTER
        MOV     ZERO_BIT,#8     ; Line number 8 is active.
EIGHT:  RET
```

270068-19

Figure 18. Subroutine SCAN Determines Which of 8 Bits in LINE is Zero

Notice that RESPONSE_TO_KEY_CLOSURE does not change the Accumulator, the PSW, nor any of the registers R0 through R7. Neither do SCAN or DEBOUNCE_DELAY.

What we come out with then is a one-byte key address (ADDRESS) which identifies the pressed key. The key's scan line number is in the upper nibble of ADDRESS, and its receive line number is in the lower nibble. ADDRESS can be used in a look-up table to generate a key code to transmit to a host computer, and/or to a display device.

The keyboard interrupt itself must be edge-triggered, rather than level-activated, so that the interrupt routine is invoked when a key is pressed, and is not constantly being repeated as long as the key is held down. In edge-triggered mode, the on-chip hardware clears the interrupt flag (EX0, in this case) as the service routine is being vectored to. In this application, however, contact bounce will cause several more edges to occur after the service routine has been vectored to, during the DEBOUNCE_DELAY routine. Consequently it is necessary to clear EX0 again in software before executing RETI.

The debounce delay routine also takes advantage of the Idle mode. In this routine a timer must be preloaded with a value appropriate to the desired length of delay. This would be

$$\text{timer preload} = \frac{(\text{osc kHz}) \times (\text{delay time ms})}{12}$$

For example, with a 3.58 MHz oscillator frequency, a 30 ms delay could be obtained using a preload value of −8950, or DD0A, in hex digits.

In the debounce delay routine (Figure 19), the timer interrupt is enabled and set to a higher priority than the keyboard interrupt, because as we invoke Idle, the keyboard interrupt is still "in progress". An interrupt of the same priority will not be acknowledged, and will not terminate the Idle mode. With the timer interrupt set to priority 1, while the keyboard interrupt is a priority 0, the timer interrupt, when it occurs, will be acknowledged and will wake up the CPU. The timer interrupt service routine does not itself have to do anything. The service routine might be nothing more than a single RETI instruction. RETI from the timer interrupt service routine then returns execution to the debounce delay routine, which shuts down the timer and returns execution to the keyboard service routine.

DRIVING AN LCD

An LCD (Liquid Crystal Display) consists of a backplane and any number of segments or dots which will be used to form the image being displayed. Applying a voltage (nominally 4 or 5V) between any segment and the backplane causes the segment to darken. The only catch is that the polarity of the applied voltage has to be periodically reversed, or else a chemical reac-

```
DEBOUNCE_DELAY:
        MOV     TL1,#TL1_PRELOAD ; Preload low byte.
        MOV     TH1,#TH1_PRELOAD ; Preload high byte.
        SETB    ET1              ; Enable Timer 1 interrupt.
        SETB    PT1              ; Set Timer 1 interrupt to high priority.
        SETB    TR1              ; Start timer running.
        ORL     PCON,#1          ; Invoke Idle mode.
;
; The next instruction will not be executed until the delay times out.
;
        CLR     TR1              ; Stop the timer.
        CLR     PT1              ; Back to priority 0 (if desired).
        CLR     ET1              ; Disable Timer 1 interrupt (if desired).
        RET                      ; Continue keyboard scan.
```

270068-20

Figure 19. Subroutine DEBOUNCE_DELAY Puts the 80C51BH into Idle During the Delay Time

tion takes place in the LCD which causes deterioration and eventual failure of the liquid crystal.

To prevent this happening, the backplane and all the segments are driven with an AC signal, which is derived from a rectangular voltage waveform. If a segment is to be "off" it is driven by the same waveform as the backplane. Thus it is always at backplane potential. If the segment is to be "on" it is driven with a waveform that is the inverse of the backplane waveform. Thus it has about 5V of periodically changing polarity between it and the backplane.

With a little software overhead, the 80C51BH can perform this task without the need for additional LCD drivers. The only drawback is that each LCD segment uses up one port pin, and the backplane uses one more. If more than, say, two 7-segment digits are being driven, there aren't many port pins left for other tasks. Nevertheless, assuming a given application leaves enough port pins available to support this task, the considerations for driving the LCD are as follows.

Suppose, for example, it is a 2-digit display with a decimal point. One port (TENS_DIGIT) connects to the 7 segments of the tens digit plus the backplane. Another port (ONES_DIGIT) connects to a decimal point plus the 7 segments of the ones digit.

One of the 80C51BH's timers is used to mark off half-periods of the drive voltage waveform. The LCD drive waveform should have a rep rate between 30 and 100 Hz, but it's not very critical. A half-period of 12 ms will set the rep rate to about 42 Hz. The preload/reload value to get 12 ms to rollover is the 2's complement negative of the oscillator frequency in kHz: if the oscillator frequency is 3.58 MHz, the reload value is −3580, or F204 in hex digits.

Now, the 80C51BH would normally be in Idle, to conserve power, during the time that the LCD and other tasks are not requiring servicing. When the timer rolls over it generates an interrupt, which brings the 80C51BH out of Idle. The service routine reloads the timer (for the next rollover), and inverts the logic levels of all the pins that are connected to the LCD. It might look like this:

```
LCD_DRIVE_INTERRUPT:
 MOV    TL1,#LOW( - XTAL_FREQ)
 MOV    TH1,#HIGH( - XTAL_FREQ)
 XRL    TENS_DIGIT,#0FFH
 XRL    ONES_DIGIT,#0FFH
 RETI
```

To update the display, one would use a look-up table to generate the characters. In the table, "on" segments are represented as 1s, and "off" segments as 0s. The backplane bit is represented as a 0. The quantity to be displayed is stored in RAM as a BCD value. The look-up table operates on the low nibble of the BCD value, and produces the bit pattern that is to be written to either the ones digit or the tens digit. Before the new patterns can be written to the LCD, the LCD drive interrupt has to be disabled. That is to prevent a polarity reversal from taking place between the times the two digits are written. An update subroutine is shown in Figure 20.

USING AN LCD DRIVER

As was noted, driving an LCD directly with an 80C51BH uses a lot of port pins. LCD drivers are available in CMOS to interface an 80C51BH to a 4-digit display using only 7 of the C51BH's I/O pins. Basically, the C51BH tells the LCD driver what digit is to be displayed (4 bits) and what position it is to be displayed in (2 bits), and toggles a Chip Select pin to tell the driver to latch this information. The LCD driver generates the display characters (hex digits), and takes care of the polarity reversals using its own RC oscillator to generate the timing.

Figure 25 shows an 80C51BH working with an ICM7211M to drive a 4-digit LCD, and the software that updates the display.

One could equally well send information to the LCD driver over the bus. In that case, one would set up the Accumulator with the digit select and data input bits, and execute a MOVX@ R0,A instruction. The LCD driver's chip select would be driven by the CPU's $\overline{WR}$ signal. This is a little easier in software than the direct bit manipulation shown in Figure 21. However, it uses more I/O pins, unless there is already some external memory involved. In that case, no extra pins are used up by adding the LCD driver to the bus.

RESONANT TRANSDUCERS

Analog transducers are often used to convert the value of a physical property, such as temperature, pressure, etc., to an analog voltage. These kinds of transducers then require an analog-to-digital converter to put the measurement into a form that is compatible with a digital control system. Another kind of transducer is now becoming available that encodes the value of the physical property into a signal that can be directly read by a digital control system. These devices are called resonant transducers.

Resonant transducers are oscillators whose frequency depends in a known way on the physical property being measured. These devices output a train of rectangular pulses whose repetition rate encodes the value of the quantity being measured. The pulses can in most cases be fed directly into the 80C51BH, which then measures either the frequency or period of the incoming signal, basing the measurement on the accuracy of its own clock oscillator. The 80C51BH can even do this in its sleep; that is, in Idle.

When the frequency or period measurement is completed, the C51BH wakes itself up for a very short time to perform a sanity check on the measurement and convert it in software to any scaling of the measured quantity that may be desired. The software conversion can include corrections for nonlinearities in the transducer's transfer function.

Resolution is also controlled by software, and can even be dynamically varied to meet changing needs as a situation becomes more critical. For example, in a process controller you can increase your resolution ("fine tune" the control, as it were) as the process approaches its target.

The nominal reference frequency of the output signal from these devices is in the range of 20 Hz to 500 kHz, depending on the design. Transducers are available that have a full scale frequency shift 2 to 1. The transducer operates from a supply voltage range of 3V to 20V, which means it can operate from the same supply voltage as the 80C51BH. At 5V, the transducer draws less than 5 mA (Reference 7). It can normally be connected directly to one of the C51BH's port pins, as shown in Figure 22.

FREQUENCY MEASUREMENTS

Measuring a frequency means counting pulses for a known sample time. Two timer/counters can be used, one to mark off the sample time and one to count pulses. If the frequency being counted doesn't exceed 50 kHz or so, one may equally well connect the transducer signal to one of the external interrupt pins, and count pulses in software. That frees up one timer, with very little cost in CPU time.

The count that is directly obtained is TxF, where T is the sample time and F is the frequency. The full scale

```
UPDATE_LCD:
        CLR     ET1             ; Disable LCD drive interrupt.
        MOV     DPTR,#TABLE_ADDRESS     ; Look-up table begins at TABLE_ADDRESS
        MOV     A,BCD_VALUE     ; Digits to be displayed.
        SWAP    A               ; Move tens digit to low nibble.
        ANL     A,#0FH          ; Mask off high nibble.
        MOVC    A,@A+DPTR       ; Tens digit pattern to accumulator.
        MOV     TENS_DIGIT,A    ; Update LCD tens digit.
        MOV     A,BCD_VALUE     ; Digits to be displayed.
        ANL     A,#0FH          ; Mask off tens digit.
        MOVC    A,@A+DPTR       ; Ones digit pattern to accumulator.
        MOV     C,DECIMAL_POINT ; Add decimal point to segment
        MOV     ACC.7,C         ;  pattern.  Update LCD decimal point
        MOV     ONES_DIGIT,A    ;  and ones digit.
        SETB    ET1             ; Re-enable LCD drive interrupt.
        RET
```

270068-21

Figure 20. UPDATE_LCD Routine Writes Two Digits to an LCD

range is Tx(Fmax-Fmin). For n-bit resolution

$$1\ LSB = \frac{Tx(Fmax\text{-}Fmin)}{2^n}$$

Therefore the sample time required for n-bit resolution is

$$T = \frac{2^n}{Fmax\text{-}Fmin}$$

For example, 8-bit resolution in the measurement of a frequency that varies between 7 kHz and 9 kHz would require, according to this formula, a sample time of 128 ms. The maximum acceptable frequency count would be 128 ms × 9 kHz = 1152 counts. The minimum would be 896 counts. Subtracting 896 from each frequency count (or presetting the frequency counter to −896 = 0FC80H) would allow the frequency to be reported on a scale of 0 to FF in hex digits.

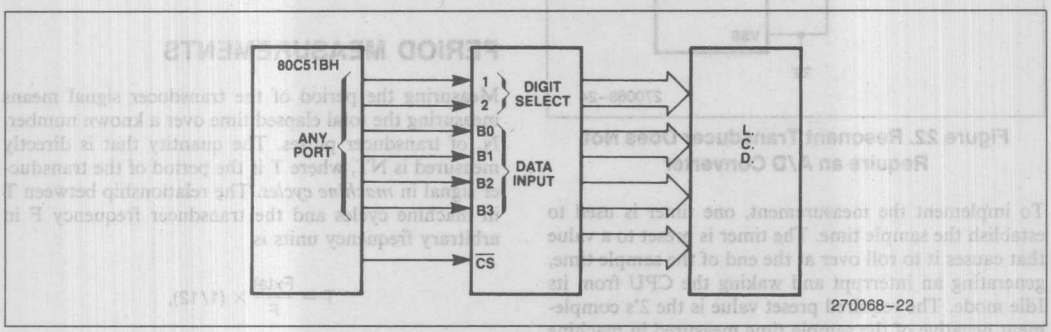


Figure 21a. Using an LCD Driver

```
UPDATE_LCD:
            MOV     A,DISPLAY_HI    ; High byte of 4-digit display.
            SETB    DIGIT_SELECT_2  ; Select leftmost digit of LCD.
            SETB    DIGIT_SELECT_1  ;   (Digit address = 11B.)
            CALL    SHIFT_AND_LOAD  ; High nibble of high byte to selected digit
            CLR     DIGIT_SELECT_1  ; Select second digit of LCD (address = 10B)
            CALL    SHIFT_AND_LOAD  ; Low nibble of high byte to selected digit.
            MOV     A,DISPLAY_LO    ; Low byte of 4-digit display.
            CLR     DIGIT_SELECT_2  ; Select third digit of LCD.
            SETB    DIGIT_SELECT_1  ;   (Digit address = 01B.)
            CALL    SHIFT_AND_LOAD  ; High nibble of low byte to selected digit.
            CLR     DIGIT_SELECT_1  ; Select fourth digit (address = 00B).
            CALL    SHIFT_AND_LOAD  ; Low nibble of low byte to selected digit.
            RET

SHIFT_AND_LOAD:
            RLC     A               ; MSB to carry bit (CY).
            MOV     DATA_INPUT_B3,C ; CY to Data Input pin B3.
            RLC     A               ; Next bit to CY.
            MOV     DATA_INPUT_B2,C ; CY to Data Input pin B2.
            RLC     A               ; Next bit to CY.
            MOV     DATA_INPUT_B1,C ; CY to Data Input pin B1.
            RLC     A               ; Last bit to CY.
            MOV     DATA_INPUT_B0,C ; CY to Data Input pin B0.
            CLR     CHIP_SELECT     ; Toggle Chip Select.
            SETB    CHIP_SELECT     ; 0-to-1 transition latches info.
            RET
```

270068-23

Figure 21b. UPDATE_LCD Routine Writes 4 Digits to an LCD Driver

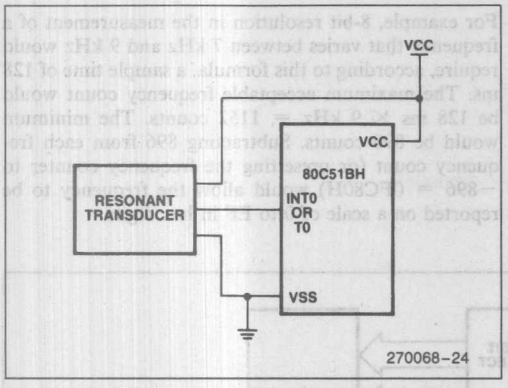


Figure 22. Resonant Transducer Does Not Require an A/D Converter

To implement the measurement, one timer is used to establish the sample time. The timer is preset to a value that causes it to roll over at the end of the sample time, generating an interrupt and waking the CPU from its Idle mode. The required preset value is the 2's complement negative of the sample time measured in machine cycles. The conversion from sample time to machine cycles is to multiply it by 1/12 the clock frequency. For example, if the clock frequency is 12 MHz, then a sample time of 128 ms is

(128 ms) × (12000 kHz)/12 = 128000 machine cycles.

Then the required preset value to cause the timer to roll over in 128 ms is

−128000 = FE0C00, in hex digits.

Note that the preset value is 3 bytes wide whereas the timer is only 2 bytes wide. This means the timer must be augmented in software in the timer interrupt routine to three bytes. The 80C51BH has a DJNZ instruction (decrement and jump if not zero) that makes it easier to code the third timer byte to count down instead of up. If the third timer byte counts down, its reload value is the 2's complement of what it would be for an up-counter. For example, if the 2's complement of the sample time is FE0C00, then the reload value for the third timer byte would be 02, instead of FE. The timer interrupt routine might then be:

```
TIMER_INTERRUPT_ROUTINE:
        DNJZ    THIRD_TIMER_BYTE,OUT
        MOV     TL0,#0
        MOV     TH0,#0CH
        MOV     THIRD_TIMERBYTE,#2
        MOV     FREQUENCY,COUNTER_LO
;Preset COUNTER to -896:
        MOV     COUNTER_LO,#80H
        MOV     COUNTER_HI,#0FCH
OUT:    RETI
```

At this point the value of the frequency of the transducer signal, measured to 8 bit resolution, is contained in FREQUENCY. Note that the timer can be reloaded on the fly. Note too that the timer can be reloaded on the fly. Note too that for 8-bit resolution only the low byte of the frequency counter needs to be read, since the high byte is necessarily 0. However, one may want to test the high byte to ensure that it is zero, as a sanity check on the data. Both bytes, of course must be reloaded.

PERIOD MEASUREMENTS

Measuring the period of the transducer signal means measuring the total elapsed time over a known number, N, of transducer pulses. The quantity that is directly measured is NT, where T is the period of the transducer signal in *machine cycles*. The relationship between T in machine cycles and the transducer frequency F in arbitrary frequency units is

$$T = \frac{Fxtal}{F} \times (1/12),$$

where Fxtal is the 80C51BH clock frequency, in the same units as F.

The full scale range then is Nx (Tmax − Tmin). For n-bit resolution.

$$1\ LSB = \frac{Ns(Tmax\text{-}Tmin)}{2^n}.$$

Therefore the number of periods over which the elapsed time should be measured is

$$N = \frac{2^n}{Tmax\text{-}Tmin}$$

However, N must also be an integer. It is logical to evaluate the above formula (don't forget Tmax and Tmin have to be in machine cycles) and select for N the next higher integer. This selection gives a period measurement that has somewhat more than n-bit resolution, but it can be scaled back if desired.

For example, suppose we want 8-bit resolution in the measurement of the period of a signal whose frequency varies from 7.1 kHz to 9 kHz. If the clock frequency is 12 MHz, then Tmax is (12000 kHz/7.1 kHz) x (1/12) = 141 machine cycles. Tmin is 111 machine cycles. The required value for N, then, is 256/(141-111) = 8.53 periods, according to the formula. Using N = 9 periods will give a maximum NT value of 141 x 9 = 1269 machine cycles. The minimum NT will be 111 × 9 = 999 machine cycles. A lookup table can be used to

scale these values back to a range of 0 to 255, giving precisely the 8-bit resolution desired.

To implement the measurement, one timer is used to measure the elapsed time, NT. The transducer is connected to one of the external interrupt pins, and this interrupt is configured to the transition-activated mode. In the transition-activated mode every 1-to-0 transition in the transducer output will generate an interrupt. The interrupt routine counts transducer pulses, and when it gets to the predetermined N, it reads and clears the timer. For the specific example cited above, the interrupt routine might be:

```
INTERRUPT_RESPONSE:
            DJNZ    N,OUT
            MOV     N,#9
            CLR     EA
            CLR     TR1
            MOV     NT_LO,TL1
            MOV     NT_HI,TH1
            MOV     TL1,#9
            MOV     TH1,#0
            SETB    TR1
            SETB    EA
            CALL    LOOKUP_TABLE
OUT:        RETI
```

In this routine a pulse counter N is decremented from its preset value, 9, to zero. When the counter gets to zero it is reloaded to 9. Then all interrupts are blocked for a short time while the timer is read and cleared. The timer is stopped during the read and clear operations, so "clearing" it actually means presetting it to 9, to make up for the 9 machine cycles that are missed while the timer is stopped.

The subroutine LOOKUP_TABLE is used to scale the measurement back to the desired 8-bit resolution. It can also include built-in corrections for errors or nonlinearities in the transducer's transfer function.

The subroutine uses the MOVC A, @ A + DPTR instruction to access the table, which contains 270 entries commencing at the 16-bit address referred to as TABLE. The subroutine must compute the address of the table entry that corresponds to the measured value of NT. This address is

$$DPTR = TABL + NT - NTMIN,$$

where NTMIN = 999, in this specific example.

```
LOOKUP_TABLE:
            PUSH  ACC
            PUSH  PSW
            MOV   A,#LOW(TABLE-NTMIN)
            ADD   A,NT_LO
            MOV   DPL,A
            MOV   A,#HIGH(TABLE-NMTIN)
            ADDC  A,NT_HI
            MOV   DPH,A
            CLR   A
            MOVC  A,@A+DTPR
            MOV   PERIOD,A
            POP   PSW
            POP   ACC
            RET
```

At this point the value of the period of the transducer signal, measured to 8 bit resolution, is contained in PERIOD.

PULSE WIDTH MEASUREMENTS

The 80C51BH timers have an operating mode which is particularly suited to pulse width measurements, and will be useful in these applications if the transducer signal has a fixed duty cycle.

In this mode the timer is turned on by the on-chip circuitry in response to an input high at the external interrupt pin, and off by an input low, and it can do this while the 80C51BH is in Idle. (The "GATE" mode of timer operation is described in the Intel Microcontroller Handbook.) The external interrupt itself can be enabled, so the same 1-to-0 transition from the transducer that turns off the timer also generates an interrupt. The interrupt routine then reads and resets the timer.

The advantage of this method is that the transducer signal has direct access to the timer gate, with the result that variations in interrupt response time have no effect on the measurement.

Resonant transducers that are designed to fully exploit the GATE mode have an internal divide-by-N circuit that fixes the duty cycle at 50% and lowers the output frequency to the range of 250 to 500 Hz (to control RFI). The transfer function between transducer period and measurand is approximately linear, with known and repeatable error functions.

HMOS/CHMOS Interchangeability

The CHMOS version of the 8051 is architecturally identical with the HMOS version, but there are nevertheless some important differences between them which the designer should be aware of. In addition, some applications require interchangeability between HMOS and CHMOS parts. The differences that need to be considered are as follows:

External Clock Drive: To drive the HMOS 8051 with an external clock signal, one normally grounds the XTAL1 pin and drives the XTAL2 pin. To drive the CHMOS 8051 with an external clock signal, one must drive the XTAL1 pin and leave the XTAL2 pin unconnected. The reason for the difference is that in the

internal clocking circuits, whereas in the CHMOS version it is the XTAL1 pin that drives the internal clocking circuits.

There are several ways to design an external clock drive to work with both types. For low clock frequencies (below 6 MHz), the HMOS 8051 can be driven in the same way as the CHMOS version, namely, through XTAL1 with XTAL2 unconnected. Another way is to drive both XTAL1 **and** XTAL2; that is, drive XTAL1 and use and external inverter to derive from XTAL1 a signal with which to drive XTAL2.

In either case, a 74HC or 74HCT circuit makes an excellent driver for XTAL1 and/or XTAL2, because neither the HMOS nor the CHMOS XTAL pins have TTL-like input logic levels.

Unused Pins: Unused pins of Ports 1, 2 and 3 can be ignored in both HMOS and CHMOS designs. The internal pullups will put them into a defined state. Unused Port 0 pins in 8051 applications can be ignored, even if they're floating. But in 80C51BH applications, these pins should not be left afloat. They can be externally pulled up or down, or they can be internally pulled down by writing 0s to them.

8031/80C31BH designs may or may not need pullups on Port 0. Pullups aren't needed for program fetches, because in bus operations the pins are actively pulled high or low by either the 8031 or the external program memory. But they are needed for the CHMOS part if the Idle or Power Down mode is invoked, because in these modes Port 0 floats.

Logic Levels: If V_{CC} is between 4.5V and 5.5V, an input signal that meets the HMOS 8051's input logic levels will also meet the CHMOS 80C51BH's input logic levels (except for XTAL1/XTAL2 and RST). For the same V_{CC} condition, the CHMOS device will reach or surpass the output logic levels of the HMOS device. The HMOS device will not necessarily reach the output logic levels of the CHMOS device. This is an important consideration if HMOS/CHMOS interchangeability must be maintained in an otherwise CMOS system.

HMOS 8051 outputs that have internal pullups (Ports 1, 2, and 3) "typically" reach 4V or more if I_{OH} is zero, but not fast enough to meet timing specs. Adding an external pullup resistor will ensure the logic level, but still not the timing, as shown in Figure 23. If timing is an issue, the best way to interface HMOS to CMOS is through a 74HCT circuit.

Idle and Power Down: The Idle and Power Down modes exist only on the CHMOS devices, but if one

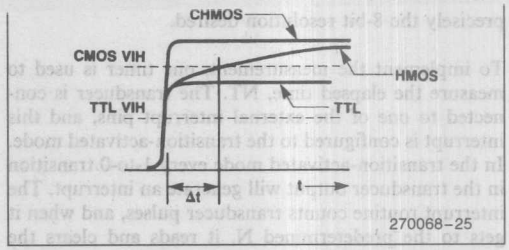


Figure 23. 0-to-1 Transition Shows Unspec'd Delay (Δt) in HMOS to 74HC Logic

wishes to preserve the capability of interchanging HMOS and CHMOS 8051s the software has to be designed so that the HMOS parts will respond in an acceptable manner when a CHMOS reduced power mode is invoked.

For example, an instruction that invokes Power Down can be followed by a "JMP $":

```
CLR   EA
ORL   PCON,#2
JMP   $
```

The CHMOS and HMOS parts will respond to this sequence of code differently. The CHMOS part, going into a normal CHMOS Power Down Mode, will stop fetching instructions until it gets a hardware reset. The HMOS part will go through the motions of executing the ORL instruction, and then fetch the JMP instruction. It will continue fetching and executing JMP $ until hardware reset.

Maintaining HMOS/CHMOS 8051 interchangeability in response to Idle requires more planning. The HMOS part will not respond to the instruction that puts the CHMOS part into Idle, so that instruction needs to be followed by a software idle. This would be an idling loop which would be terminated by the same conditions that would terminate the CHMOS's hardware Idle. Then when the CHMOS device goes into Idle, the HMOS version executes the idling loop, until either a hardware reset or an enabled interrupt is received. Now if Idle is terminated by an interrupt, execution for the CHMOS device will proceed after RETI from the instruction following the one that invoked Idle. The instruction following the one that invoked Idle is the idling loop that was inserted for the HMOS device. At this point, both the HMOS and CHMOS devices must be able to fall through the loop to continue execution.

One way to achieve the desired effect is to define a "fake" Idle flag, and set it just before going into Idle. The instruction that invoked Idle is followed by a software idle:

```
SETB    IDLE
ORL     PCON,#1
JB      IDLE,$
```

Now the interrupt that terminates the CHMOS's Idle must also break the software idle. It does so by clearing the "Idle" bit:

```
...
CLR     IDLE
RETI
```

Note too that the PCON register in the HMOS 8051 contains only one bit, SMOD, whereas the PCON register in CHMOS contains SMOD plus four other bits. Two of those other bits are general purpose flags. Maintaining HMOS/CHMOS interchangeability requires that these flags not be used.

REFERENCES

1. Pawlowski, Moroyan, Alnether, "Inside CMOS Technology," *BYTE magazine,* Sept., 1983. Available as Article Reprint AR-302.
2. Kokkonen, Pashley, "Modular Approach to C-MOS Technology Tailors Process to Application," *Electronics*, May, 1984. Available as Article Reprint AR-332.
3. Williamson, T., *Designing Microcontroller Systems for Electrically Noisy Environments,* Intel Application Note AP-125, Feb. 1982.
4. Williamson, T., "PC Layout Techniques for Minimizing Noise," *Mini-Micro Southeast,* Session 9, Jan., 1984.
5. Alnether, J., *High Speed Memory System Design Using 2147H, Intel Application Note AP-74, March 1980.*
6. Ott, H., "Digital Circuit Grounding and Interconnection," *Proceedings of the IEEE Symposium on Electromagnetic Compatibility,* pp. 292–297, Aug. 1981.
7. *Digital Sensors by Technar,* Technar Inc., 205 North 2nd Ave., Arcadia, CA 91006.

APPLICATION NOTE

AP-281

UPI-452 Accelerates iAPX 286 Bus Performance

CHRISTOPHER SCOTT
TECHNICAL MARKETING ENGINEER
INTEL CORPORATION

July 1986

Order Number: 292018-001

INTRODUCTION

The UPI-452 targets the leading problem in peripheral to host interfacing, the interface of a slow peripheral with a fast Host or "bus utilization". The solution is data buffering to reduce the delay and overhead of transferring data between the Host microprocessor and I/O subsystem. The Intel CMOS UPI-452 solves this problem by combining a sophisticated programmable FIFO buffer and a slave interface with an MSC-51 based microcontroller.

The UPI-452 is Intel's newest Universal Peripheral Interface family member. The UPI-452 FIFO buffer enables Host—peripheral communications to be through streams or bursts of data rather than by individual bytes. In addition the FIFO provides a means of embedding commands within a stream or block of data. This enables the system designer to manage data and commands to further off-load the Host.

The UPI-452 interfaces to the iAPX 286 microprocessor as a standard Intel slave peripheral device. READ, WRITE, CS and address lines from the Host are used to access all of the Host addressable UPI-452 Special Function Registers (SFR).

The UPI-452 combines an MSC-51 microcontroller, with 256 bytes of on-chip RAM and 8K bytes of EPROM/ROM, twice that of the 80C51, a two channel DMA controller and a sophisticated 128 byte, two channel, bidirectional FIFO in a single device. The UPI-452 retains all of the 80C51 architecture, and is fully compatible with the MSC-51 instruction set.

This application note is a description of an iAPX 286 to UPI-452 slave interface. Included is a discussion of the respective timings and design considerations. This application note is meant as a supplement to the UPI-452 Advance Data Sheet. The user should consult the data sheet for additional details on the various UPI-452 functions and features.

UPI-452 iAPX 286 SYSTEM CONFIGURATION

The interface described in this application note is shown in Figure 1, iAPX 286 UPI-452 System Block Diagram. The iAPX 286 system is configured in a local bus architecture design. DMA between the Host and the UPI-452 is supported by the 82258 Advanced DMA Controller. The Host microprocessor accesses all UPI-452 externally addressable registers through address decoding (see Table 3, UPI-452 External Address Decoding). The timings and interface descriptions below are given in equation form with examples of specific calculations. The goal of this application note is a set of interface analysis equations. These equations are the tools a system designer can use to fully utilize the features of the UPI-452 to achieve maximum system performance.

HOST-UPI-452 FIFO SLAVE INTERFACE

The UPI-452 FIFO acts as a buffer between the external Host 80286 and the internal CPU. The FIFO allows the Host - peripheral interface to achieve maximum decoupling of the interface. Each of the two FIFO channels is fully user programmable. The FIFO buffer ensures that the respective CPU, Host or internal CPU, receives data in the same order as transmitted. Three slave bus interface handshake methods are supported by the UPI-452; DMA, Interrupt and Polled.

The interface between the Host 80286 and the UPI-452 is accomplished with a minimum of signals. The 8 bit data bus plus READ, WRITE, CS, and A0-2 provide access to all of the externally addressable UPI-452 registers including the two FIFO channels. Interrupt and DMA handshaking pins are tied directly to the interrupt controller and DMA controller respectively.

DMA transfers between the Host and UPI-452 are controlled by the Host processors DMA controller. In the example shown in Figure 1, the Host DMA controller is the 82258 Advanced DMA Controller. An internal DMA transfer to or from the FIFO, as well as between other internal elements, is controlled by the UPI-452 internal DMA processor. The internal DMA processor can also transfer data between Input and Output FIFO channels directly. The description that follows details the UPI-452 interface from both the Host processor's and the UPI-452's internal CPU perspective.

One of the unique features of the UPI-452 FIFO is its ability to distinguish between commands and data embedded in the same data block. Both interrupts and status flags are provided to support this operation in either direction of data transfer. These flags and interrupts are triggered by the FIFO logic independent of, and transparent to either the Host or internal CPUs. Commands embedded in the data block, or stream, are called Data Stream Commands.

Programmable FIFO channel Thresholds are another unique feature of the UPI-452. The Thresholds provide for interrupting the Host only when the Threshold number of bytes can be read or written to the FIFO buffer. This further decouples the Host UPI-452 interface by relieving the Host of polling the buffer to determine the number of bytes that can be read or written. It also reduces the chances of overrun and underrun errors which must be processed.

The UPI-452 also provides a means of bypassing the FIFO, in both directions, for an immediate interrupt of either the Host or internal CPU. These commands are called Immediate Commands. A complete description of the internal FIFO logic operation is given in the FIFO Data Structure section.

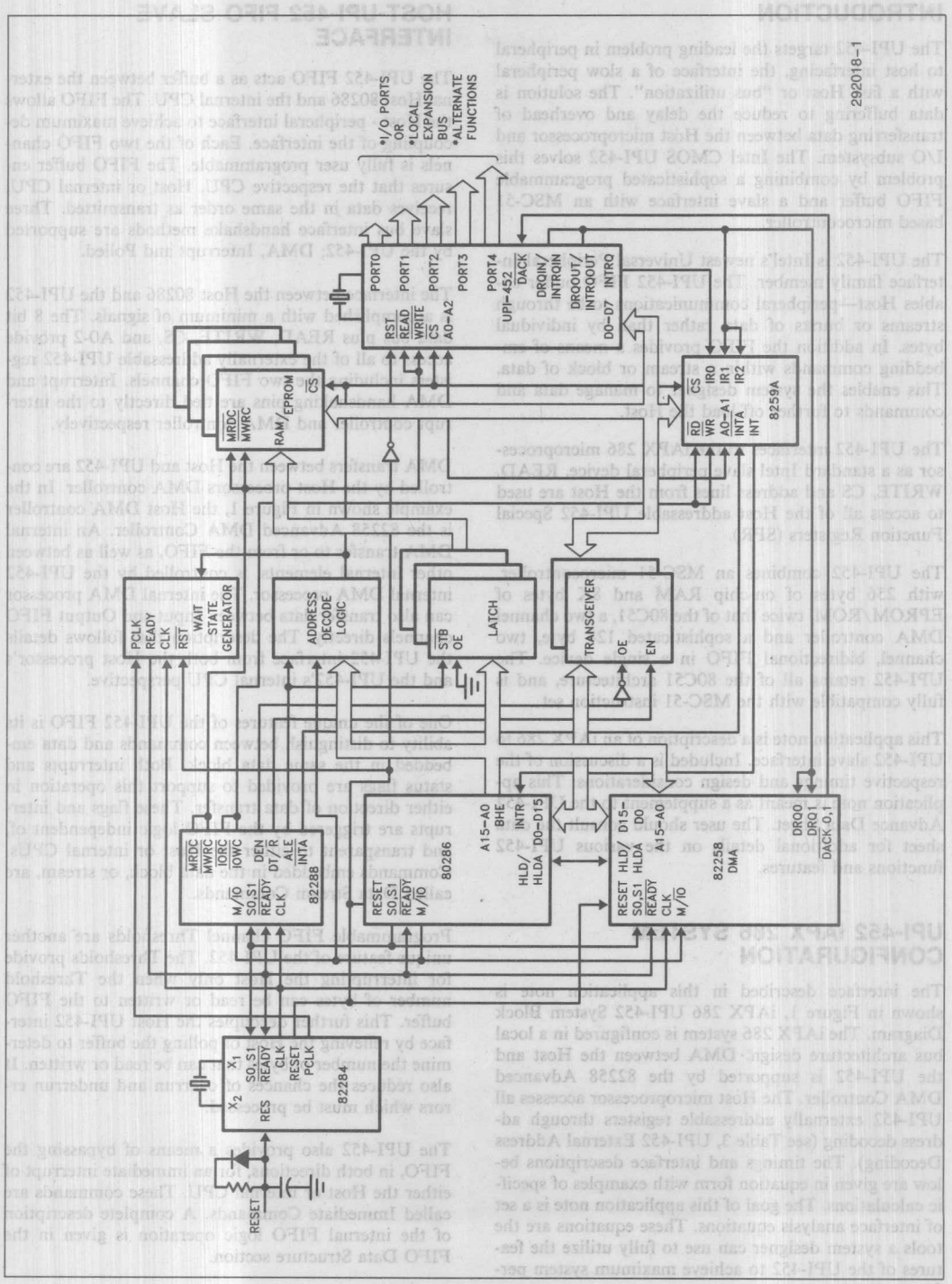


Figure 1. iAPX 286 UPI-452 System Block Diagram

UPI-452 INITIALIZATION

The UPI-452 at power-on reset automatically performs a minimum initialization of itself. The UPI-452 notifies the Host that it is in the process of initialization by setting a Host Status SFR bit. The user UPI-452 program must release the UPI-452 from initialization for the FIFO to be accessible by the Host. This is the minimum Host to UPI-452 initialization sequence. All further initialization and configuration of the UPI-452, including the FIFO, is done by the internal CPU under user program control. No interaction or programming is required by the Host 80286 for UPI-452 initialization.

At power-on reset the UPI-452 automatically enters FIFO DMA Freeze Mode by resetting the Slave Control (SLCON) SFR FIFO DMA Freeze/Normal Mode bit to FIFO DMA Freeze Mode (FRZ = "0"). This forces the Slave Status (SSTAT) and Host Status (HSTAT) SFR FIFO DMA Freeze/Normal Mode bits to FIFO DMA Freeze Mode In Progress. FIFO DMA Freeze Mode allows the FIFO interface to be configured, by the internal CPU, while inhibiting Host access to the FIFO.

The MODE SFR is forced to zero at reset. This disables, (tri-states) the DRQIN/INTRQIN, DRQOUT/INTRQOUT and INTRQ output pins. INTRQ is inhibited from going active to reflect the fact that a Host Status SFR bit, FIFO DMA Freeze Mode, is active. If the MODE SFR INTRQ configure bit is enabled (='1'), before the Slave Control and Host Status SFR FIFO DMA Freeze/Normal Mode bit is set to Normal Mode, INTRQ will go active immediately.

The first action by the Host following reset is to read the UPI-452 Host Status SFR Freeze/Normal Mode bit to determine the status of the interface. This may be done in response to a UPI-452 INTRQ interrupt, or by polling the Host Status SFR. Reading the Host Status SFR resets the INTRQ line low.

Any of the five FIFO interface SFRs, as well as a variety of additional features, may be programmed by the internal CPU following reset. At power-on reset, the five FIFO Special Function Registers are set to their default values as listed in Table 1. All reserved location bits are set to one, all other bits are set to zero in these three SFRs. The FIFO SFRs listed in Table 1 can be programmed only while the UPI-452 is in FIFO DMA Freeze Mode. The balance of the UPI-452 SFRs default values and descriptions are listed in the UPI-452 Advance Data Sheet in the Intel Microsystems Component Handbook Volume II and Microcontroller Handbook.

The above sequence is the minimum UPI-452 internal initialization required. The last initialization instruction must always set the UPI-452 to Normal Mode. This causes the UPI-452 to exit Freeze Mode and enables Host read/write access of the FIFO. The internal CPU sets the Slave Control (SLCON) SFR FIFO DMA Freeze/Normal Mode (FRZ) bit high (=1) to activate Normal Mode. Ths causes the Slave Status (SSTAT) and Host Status (HSTAT) SFR FIFO DMA Freeze Mode bits to be set to Normal Mode. Table 2, UPI-452 Initialization Event Sequence Example, shows a summary of the initialization events described above.

Table 1. FIFO Special Function Register Default Values

SFR Name	Label	Reset Value
Channel Boundary Pointer	CBP	40H/64D
Output Channel Read Pointer	ORPR	40H/64D
Output Channel Write Pointer	OWPR	40H/64D
Input Channel Read Pointer	IRPR	00H/0D
Input Channel Write Pointer	IWPR	00H/0D
Input Threshold	ITH	00H/0D
Output Threshold	OTH	01H/1D

Table 2. UPI-452 Initialization Event Sequence Example

Event Description	SFR/bit
Power-on Reset	
UPI-452 forces FIFO DMA Freeze Mode (Host access to FIFO inhibited)	SLCON FRZ = 0
UPI-452 forces Slave Status and Host Status SFR to FIFO DMA Freeze Mode In Progress	SSTAT SST5 = 0 HSTAT HST1 = 1
UPI-452 forces all SFRs, including FIFO SFRs, to default values.	
* UPI-452 user program enables INTRQ, INTRQ goes active, high	MODE MD4 = 1
* Host READ's UPI-452 Host Status (HSTAT) SFR to determine interrupt source, INTRQ goes low	
* UPI-452 user program initializes any other SFRs; FIFO, Interrupts, Timers/Counters, etc.	
User program sets Slave Control SFR to Normal Mode (Host access to FIFO enabled)	SLCON FRZ = 1
UPI-452 forces Slave and Host Status SFRs bits to Normal Operation	SSTAT SST5 = 1 HSTAT HST1 = 0
* Host polls Host Status SFR to determine when it can access the FIFO - or - * Host waits for UPI-452 Request for Service interrupt to access FIFO	

* user option

FIFO DATA STRUCTURES

Overview

The UPI-452 provides three means of communication between the Host microprocessor and the UPI-452 in either direction;

Data
Data Stream Commands
Immediate Commands

Data and Data Stream Commands (DSC) are transferred between the Host and UPI-452 through the UPI-452 FIFO buffer. The third, Immediate Commands, provides a means of bypassing the FIFO entirely. These three data types are in addition to direct access by either Host or Internal CPU of dedicated Status and Control Special Function Registers (SFR).

The FIFO appears to both the Host 80286 and the internal CPU as 8 bits wide. Internally the FIFO is logically nine bits wide. The ninth bit indicates whether the byte is a data or a Data Stream Command (DSC) byte; 0 = data, 1 = DSC. The ninth bit is set by the FIFO logic in response to the address specified when writing to the FIFO by either Host or internal CPU. The FIFO uses the ninth bit to condition the UPI-452 interrupts and status flags as a byte is made available for a Host or internal CPU read from the FIFO. Figures 2 and 3 show the structure of each FIFO channel and the logical ninth bit.

It is important to note that both data and DSCs are actually entered into the FIFO buffer (see Figures 2 and 3). External addressing of the FIFO determines the state of the internal FIFO logic ninth bit. Table 3 shows the UPI-452 External Address Decoding used by the Host and the corresponding action. The internal CPU interface to the FIFO is essentially identical to the external Host interface. Dedicated internal Special Function Registers provide the interface between the FIFO, internal CPU and the internal two channel DMA processor. FIFO read and write operations by the Host and internal CPU are interleaved by the UPI-452 so they appear to be occurring simultaneously.

The ninth bit provides a means of supporting two data types within the FIFO buffer. This feature enables the Host and UPI-452 to transfer both commands and data while maintaining the decoupled interface a FIFO buffer provides. The logical ninth bit provides both a means of embedding commands within a block of data and a means for the internal CPU, or external Host, to discriminate between data and commands. Data or DSCs may be written in any order desired. Data Stream Commands can be used to structure or dispatch the data by defining the start and end of data blocks or packets, or how the data following a DSC is to be processed.

A Data Stream Command (DSC) acts as an internal service routine vector. The DSC generates an interrupt to a service routine which reads the DSC. The DSC byte acts as an address vector to a user defined service routine. The address can be any program or data memory location with no restriction on the number of DSCs or address boundaries.

A Data Stream Command (DSC) can also be used to clear data from the FIFO or "FLUSH" the FIFO. This is done by appending a DSC to the end of a block of data entered in the FIFO which is less than the programmed threshold number of bytes. The DSC will cause an interrupt, if enabled, to the respective receiving CPU. This ensures that a less than Threshold number of bytes in the FIFO will be read. Two conditions force a Request for Service interrupt, if enabled, to the Host. The first is due to a Threshold number of bytes having been written to the FIFO Output channel; the second is if a DSC is written to the Output FIFO channel. If less than the Threshold number of bytes are written to the Output FIFO channel, the Host Status SFR flag will not be set, and a Request for Service interrupt will not be generated, if enabled. By appending a DSC to end of the data block, the FIFO Request for Service flag and/or interrupt will be generated.

An example of a FIFO Flush application is a mass storage subsystem. The UPI-452 provides the system interface to a subsystem which supports tape and disk storage. The FIFO size is dynamically changed to provide the maximum buffer size for the direction of transfer. Large data blocks are the norm in this application. The FIFO Flush provides a means of purging the FIFO of the last bytes of a transfer. This guarantees that the block, no matter what its size, will be transmitted out of the FIFO.

Immediate Commands allow more direct communication between the Host processor and the UPI-452 by bypassing the FIFO in either direction. The Immediate Command IN and OUT SFRs are two more unique address locations externally and internally addressable. Both DSCs and Immediate Commands have internal interrupts and interrupt priorities associated with their operation. The interrupts are enabled or disabled by setting corresponding bits in the Slave Control (SLCON), Interrupt Enable (IEC), Interrupt Priority (IPC) and Interrupt Enable and Priority (IEP) SFRs. A detailed description of each of these may be found in the UPI-452 Advance Information Data Sheet.

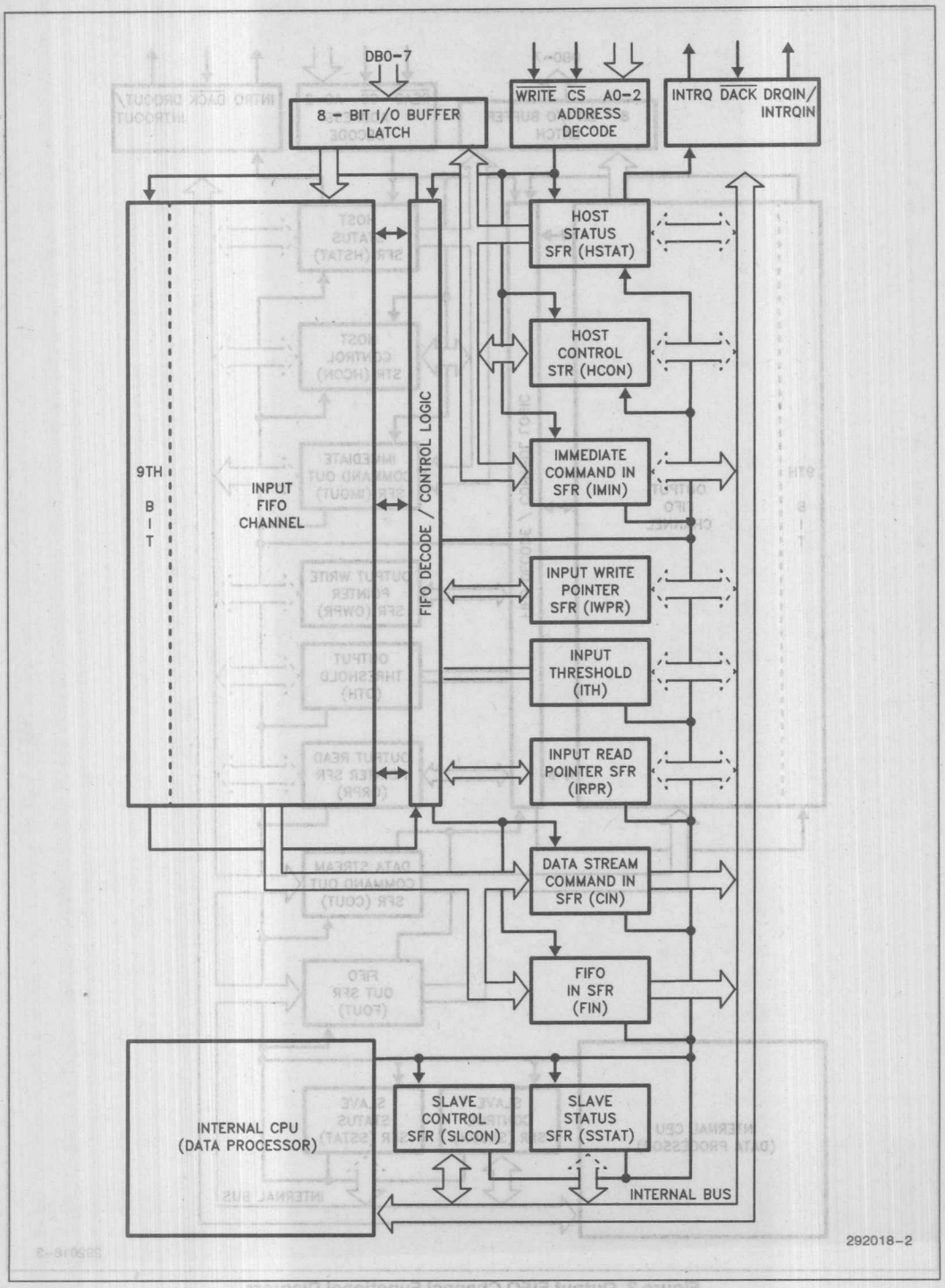


Figure 2. Input FIFO Channel Functional Diagram

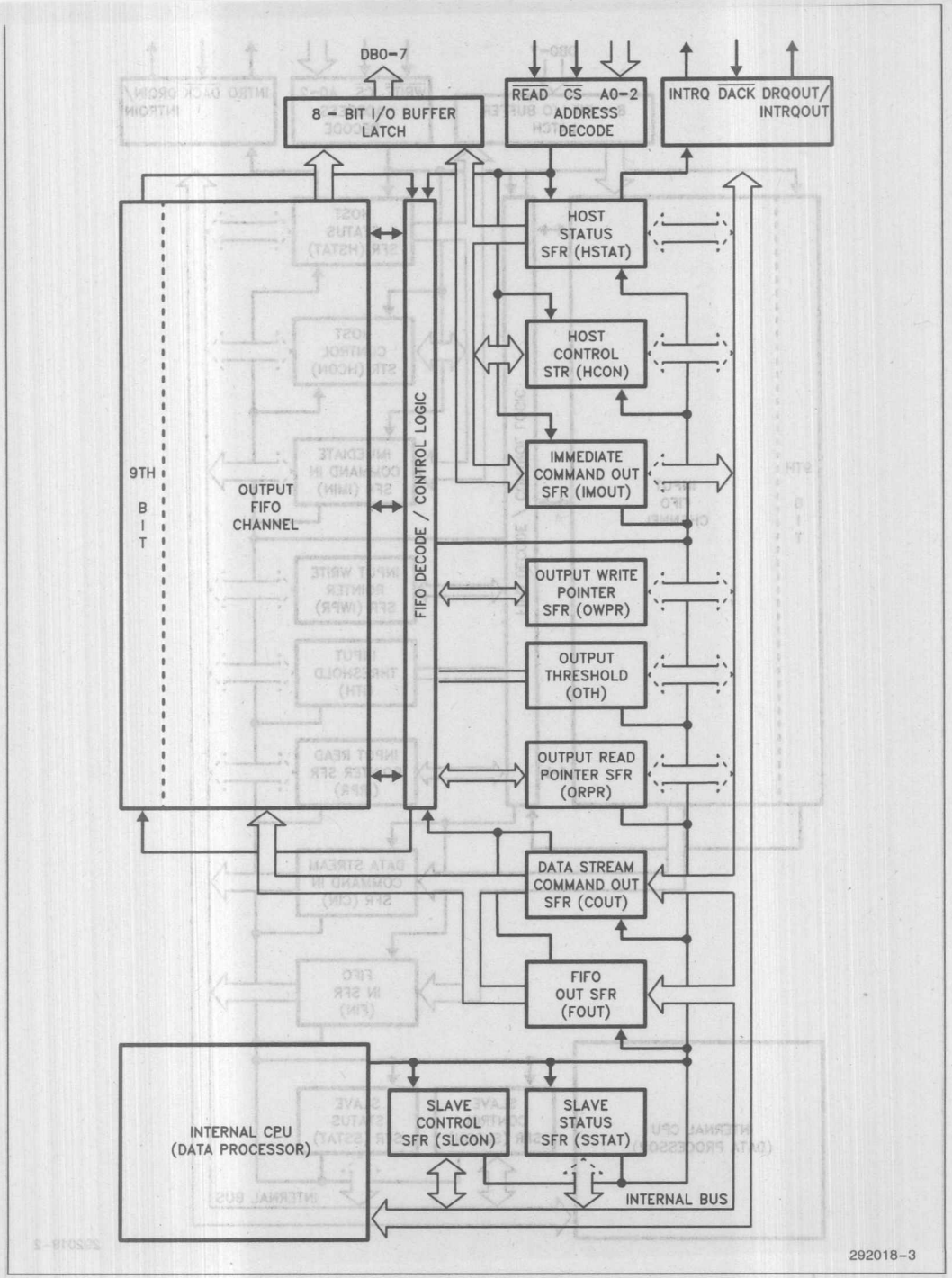


Figure 3. Output FIFO Channel Functional Diagram

Table 3. UPI-452 External Address Decoding

DACK	CS	A2	A1	A0	READ	WRITE
1	1	X	X	X	No Operation	No Operation
1	0	0	0	0	Data or DMA from Output FIFO Channel	Data or DMA to Input FIFO Channel
1	0	0	0	1	Data Stream Command from Output FIFO Channel	Data Stream Command to Input FIFO Channel
1	0	0	1	0	Host Status SFR Read	Reserved
1	0	0	1	1	Host Control SFR Read	Host Control SFR Write
1	0	1	0	0	Immediate Command SFR Read	Immediate Command SFR Write
1	0	1	1	X	Reserved	Reserved
0	X	X	X	X	DMA Data from Output FIFO Channel	DMA Data to Input FIFO Channel

Below is a detailed description of each FIFO channel's operation, including the FIFO logic response to the ninth bit, as a byte moves through the channel. The description covers each of the three data types for each channel. The details below provide a picture of the various FIFO features and operation. By understanding the FIFO structure and operation the user can optimize the interface to meet the requirements of an individual design.

OUTPUT CHANNEL

This section covers the data path from the internal CPU to the HOST. Data Stream Command or Immediate Command processing during Host DMA Operations is covered in the DMA section.

UPI-452 Internal Write to the FIFO

The internal CPU writes data and Data Stream Commands into the FIFO through the FIFO OUT (FOUT) and Command OUT (COUT) SFRs. When a Threshold number of bytes has been written, the Host Status SFR Output FIFO Request for Service bit is set and an interrupt, if enabled, is generated to the Host. Either the INTRQ or DRQOUT/INTRQOUT output pins can be used for this interrupt as determined by the MODE and Host Control (HCON) SFR setting. The Host responds to the Request for Service interrupt by reading the Host Status (HSTAT) SFR to determine the source of the interrupt. The Host then reads the Threshold number of bytes from the FIFO. The internal CPU may continue to write to the FIFO during the Host read of the FIFO Output channel.

Data Stream Commands may be written to the Output FIFO channel at any time during a write of data bytes. The write instruction need only specify the Command Out (COUT) SFR in the direct register instruction used. Immediate Commands may also be written at any time to the Immediate Command OUT (IMOUT) SFR. The Host reads Immediate Commands from the Immediate Command OUT (IMOUT).

The internal CPU can determine the number of bytes to write to the FIFO Output channel in one of three ways. The first, and most efficient, is by utilizing the internal DMA processor which will automatically manage the writing of data to avoid Underrun or Overrun Errors. The second is for the internal CPU to read the Output FIFO channels Read and Write Pointers and compare their values to determine the available space. The third method for determining the available FIFO space is to always write the programmed channel size number of bytes to the Output FIFO. This method would use the Overrun Error flag and interrupt to halt FIFO writing whenever the available space was less than the channel size. The interrupt service routine could read the channel pointers to determine or monitor the available channel space. The time required for the internal CPU to write data to the Output FIFO channel is a function of the individual instruction cycle time and the number of bytes to be written.

Host Read from the FIFO

The Host reads data or Data Stream Commands (DSC) from the FIFO in response to the Host Status (HSTAT) SFR flags and interrupts, if enabled. All Host read operations access the same UPI-452 internal I/O Buffer Latch. At the end of the previous Host FIFO read cycle a byte is loaded from the FIFO into the I/O Buffer Latch and Host Status (HSTAT) SFR bit 5 is set or cleared (1 = DSC, 0 = data) to reflect the state of the byte's FIFO ninth bit. If the FIFO ninth bit is set (= 1) indicating a DSC, an interrupt is generated to the external Host via INTRQ pin or INTRQIN/INTRQOUT pins as determined by Host Control (HCON) SFR bit 1. The Host then reads the Host Status (HSTAT) SFR to determine the source of the interrupt.

The most efficient Host read operation of the FIFO Output channel is through the use of Host DMA. The UPI-452 fully supports external DMA handshaking. The MODE and Host Control SFRs control the configuration of UPI-452 Host DMA handshake outputs. If Host DMA is used the Threshold Request for Service interrupt asserts the UPI-452 DMA Request (DRQOUT) output. The Host DMA processor acknowledges with $\overline{\text{DACK}}$ which acts as a chip select of the FIFO channels. The DMA transfer would stop when either the threshold byte count had been read, as programmed in the Host DMA processor, or when the DRQOUT output is brought inactive by the UPI-452.

INPUT CHANNEL

This section covers the data path from the HOST to the internal CPU or internal DMA processor. The details of Data Stream Command or Immediate Command processing during internal DMA operations are covered in the DMA section below.

Host Write to the FIFO

The Host writes data and Data Stream Commands into the FIFO through the FIFO IN (FIN) and Command IN (CIN) SFRs. When a Threshold number of bytes has been read out of the Input FIFO channel by the internal CPU, the Host Status SFR Input FIFO Request for Service bit is set and an interrupt, if enabled, is generated to the Host. The Input FIFO Threshold interrupt tells the Host that it may write the next block of data into the FIFO. Either the INTRQ or DRQIN/INTRQIN output pins can be used for this interrupt as determined by the MODE and Host Control (HCON) SFR settings. The Host may continue to write to the FIFO Input channel during the internal CPU read of the FIFO. Data Stream Commands may be written to the FIFO Input channel at any time during a write of data bytes. Immediate Commands may also be written at any time to the Immediate Command IN (IMIN) SFR.

The Host also has three methods for determining the available FIFO space. Two are essentially identical to that of the internal CPU. They involve reading the FIFO Input channel pointers and using the Host Status SFR Underrun and Overrun Error flags and Request for Service interrupts these would generate, if enabled in combination. The third involves using the UPI-452 Host DMA controller handshake signals and the programmed Input FIFO Threshold. The Host would receive a Request for Service interrupt when an Input FIFO channel has a Theshold number of bytes able to be written by the Host. The Host service routine would then write the Threshold number of bytes to the FIFO.

If a Host DMA is used to write to the FIFO Input channel, the Threshold Request for Service interrupt could assert the UPI-452 DRQIN output. The Host DMA processor would assert $\overline{\text{DACK}}$ and the FIFO write would be completed by Host the DMA processor. The DMA transfer would stop when either the Threshold byte count had been written or the DRQIN output was removed by the UPI-452. Additional details on Host and internal DMA operation is given below.

Internal Read of the FIFO

At the end of an internal CPU read cycle a byte is loaded from the FIFO buffer into the FIFO IN/Command IN SFR and Slave Status (SSTAT) SFR bit 1 is set or cleared (1 = data, 0 = DSC) to reflect the state of the FIFO ninth bit. If the byte is a DSC, the FIFO ninth bit is set (= 1) and an interrupt is generated, if enabled, to the Internal CPU. The internal CPU then reads the Slave Status (SSTAT) SFR to determine the source of the interrupt.

Immediate Commands are written by the Host and read by the internal CPU through the Immediate Command IN (IMIN) SFR. Once written, an Immediate Command sets the Slave Status (SSTAT) SFR flag bit and generates an interrupt, if enabled, to the internal CPU. In response to the interrupt the internal CPU reads the Slave Status (SSTAT) SFR to determine the source of the interrupt and service the Immediate Command.

FIFO INPUT/OUTPUT CHANNEL SIZE

Host

The Host does not have direct control of the FIFO Input or Output channel sizes or configuration. The Host can, however, issue Data Stream Commands or Immediate Commands to the UPI-452 instructing the UPI-452 to reconfigure the FIFO interface by invoking FIFO DMA Freeze Mode. The Data Stream Command or Immediate Command would be a vector to a service routine which performs the specific reconfiguration.

UPI-452 Internal

The default power-on reset FIFO channel sizes are listed in the "Initialization" section and can be set only by the internal CPU during FIFO DMA Freeze Mode. The FIFO channel size is selected to achieve the optimum application performance. The entire 128 byte FIFO can be allocated to either the Input or Output channel. In this case the other channel consists of a single SFR; FIFO IN/Command IN or FIFO OUT/Command OUT SFR. Figure 4 shows a FIFO division with a portion devoted to each channel. Figure 5 shows a FIFO configuration with all 128 bytes assigned to the Output channel.

The FIFO channel Threshold feature allows the user to match the FIFO channel size and the performance of the internal and Host data transfer rates. The programmed Threshold provides an elasticity to the data transfer operation. An example is if the Host FIFO

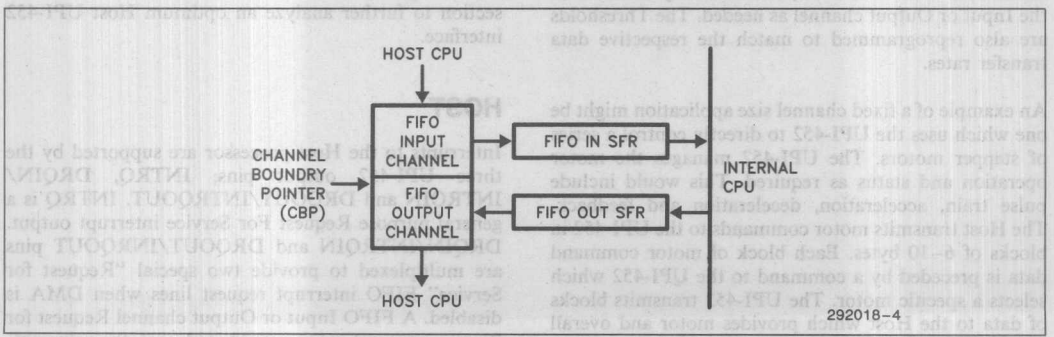


Figure 4. Full Duplex FIFO Operation

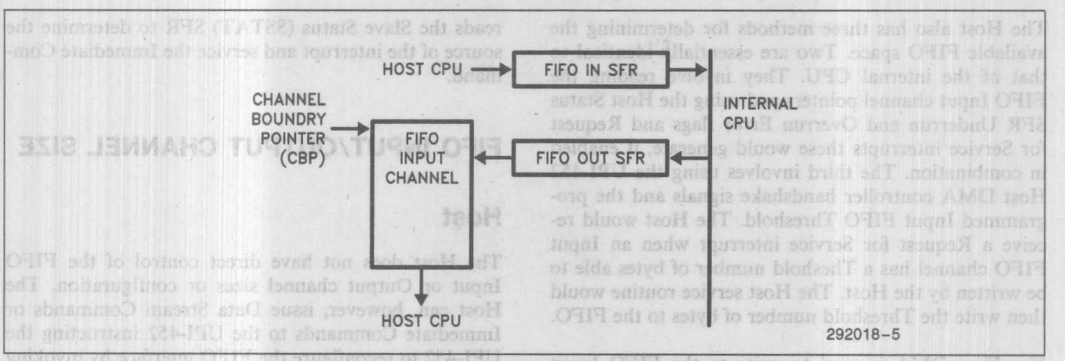


Figure 5. Entire FIFO Assigned to Output Channel

data transfer rate is twice as fast as the internal FIFO DMA data transfer rate. In this example the FIFO Input channel size is programmed to be 64 bytes and the Input channel Threshold is programmed to be 20 bytes. The Host writes the first 64 bytes to the Input FIFO. When the internal DMA processor has read 20 bytes the Threshold interrupt, or DMA request (DRQIN), is generated to signal the Host to begin writing more data to the Input FIFO channel. The internal DMA processor continues to read data from the Input channel as the Host, or Host DMA processor, writes to the FIFO. The Host can write 40 bytes to the FIFO Input channels in the time it takes for the internal DMA processor to read 20 more bytes from it. This will keep both the Host and internal DMA operating at their maximum rates without forcing one to wait for the other.

Two methods of managing the FIFO size are possible; fixed and variable channel size. A fixed channel size is one where the channel is configured at initialization and remains unchanged throughout program execution. In a variable FIFO channel size, the configuration is changed dynamically to meet the data transmission requirements as needed. An example of a variable channel size application is the mass storage subsystem described earlier. To meet the demands of a large data block transfer the FIFO size could be fully allocated to the Input or Output channel as needed. The Thresholds are also reprogrammed to match the respective data transfer rates.

An example of a fixed channel size application might be one which uses the UPI-452 to directly control a series of stepper motors. The UPI-452 manages the motor operation and status as required. This would include pulse train, acceleration, deceleration and feedback. The Host transmits motor commands to the UPI-452 in blocks of 6–10 bytes. Each block of motor command data is preceded by a command to the UPI-452 which selects a specific motor. The UPI-452 transmits blocks of data to the Host which provides motor and overall system status. The data and embedded commands structure, indicating the specific motor, is the same. In this example the default 64 bytes per channel might be adequate for both channels.

INTERRUPT RESPONSE TIMING

Interrupts enable the Host UPI-452 FIFO buffer interface and the internal CPU FIFO buffer interface to operate with a minimum of overhead on the respective CPU. Each CPU is "interrupted" to service the FIFO on an as needed basis only. In configuring the FIFO buffer Thresholds and choosing the appropriate internal DMA Mode the user must take into account the interrupt response time for both CPUs. These response times will affect the DMA transfer rates for each channel. By choosing FIFO channel Thresholds which reflect both the respective DMA transfer rate and the interrupt response time the user will achieve the maximum data throughput and system bus decoupling. This in turn will mean the overall available system bus bandwidth will increase.

The following section describes the FIFO interrupt interface to the Host and internal CPU. It also describes an analysis of sample interrupt response times for the Host and UPI-452 internal CPU. These equations and the example times shown are then used in the DMA section to further analyze an optimum Host UPI-452 interface.

HOST

Interrupts to the Host processor are supported by the three UPI-452 output pins; INTRQ, DRQIN/INTRQIN and DRQOUT/INTRQOUT. INTRQ is a general purpose Request For Service interrupt output. DRQIN/INTRQIN and DRQOUT/INRQOUT pins are multiplexed to provide two special "Request for Service" FIFO interrupt request lines when DMA is disabled. A FIFO Input or Output channel Request for Service interrupt is generated based upon the value programmed in the respective channel's Threshold SFRs; Input Threshold (ITHR), and Output Threshold

(OTHR) SFRs. Additional interrupts are provided for FIFO Underrun and Overrun Errors, Data Stream Commands, and Immediate Commands. Table 4 lists the eight UPI-452 interrupt sources as they appear in the HSTAT SFR to the Host processor.

Table 4. UPI-452 to Host Interrupt Sources

HSTAT SFR Bit	Interrupt Source
HST7	Output FIFO Underrun Error
HST6	Immediate Command Out SFR Status
HST5	Data Stream Command/Data at Output FIFO Status
HST4	Output FIFO Request for Service Status
HST3	Input FIFO Overrun Error Condition
HST2	Immediate Comamnd In SFR Status
HST1	FIFO DMA Freeze/Normal Mode Status
HST0	Input FIFO Request for Service

The interrupt response time required by the Host processor is application and system specific. In general, a typical sequence of Host interrupt response events and the approximate times associated with each are listed in Equation 1.

The example assumes the hardware configuration shown in Figure 1, iAPX 286/UPI-452 Block Diagram, with an 8259A Programmable Interrupt Controller. The timing analysis in Equation 1 also assumes the following; no other interrupt is either in process or pending, nor is the 286 in a LOCK condition. The current instruction completion time is 8 clock cycles (800 ns @ 10 MHz), or 4 bus cycles. The interrupt service routine first executes a PUSHA instruction, PUSH All General Registers, to save all iAPX 286 internal registers. This requires 19 clocks (or 2.0 μs @ 10 MHz), or 10 bus cycles (rounded to complete bus cycle). The next service routine instruction reads the UPI-452 Host Status SFR to determine the interrupt source.

It is important to note that any UPI-452 INTRQ interrupt service routine should ALWAYS mask for the Freeze Mode bit first. This will insure that Freeze Mode always has the highest priority. This will also save the time required to mask for bits which are forced inactive during Freeze Mode, before checking the Freeze Mode bit. Access to the FIFO channels by the Host is inhibited during Freeze Mode. Freeze Mode is covered in more detail below.

To initiate the interrupt the UPI-452 activates the INTRQ output. The interrupt acknowledge sequence requires two bus cycles, 400 ns (10 MHz iAPX 286), for the two INTA pulse sequence.

Equation 1. Host Interrupt Response Time

Action	Time	Bus Cycles*
Current instruction execution completion	800 ns	4
INTA sequence	400 ns	2
Interrupt service routine (time to host first $\overline{\text{READ}}$ of UPI-452)	2000 ns	10
Total Interrupt Response Time	2.3 μs	16

NOTE:
10 MHz iAPX 286 bus cycle, 200 ns each

UPI-452 Internal

The internal CPU FIFO interrupt interface is essentially identical to that of the Host to the FIFO. Three internal interrupt sources support the FIFO operation; FIFO-Slave bus Interface Buffer, DMA Channel 0 and DMA Channel 1 Requests. These interrupts provide a maximum decoupling of the FIFO buffer and the internal CPU. The four different internal DMA Modes available add flexibility to the interface.

The FIFO-Slave Bus Interface interrupt response is also similar to the Host response to an INTRQ Request for Service interrupt. The internal CPU responds to the interrupt by reading the Slave Status (SSTAT) SFR to determine the source of the interrupt. This allows the user to prioritize the Slave Status flag response to meet the users application needs.

The internal interrupt response time is dependent on the current instruction execution, whether the interrupt is enabled, and the interrupt priority. In general, to finish execution of the current instruction, respond to the interrupt request, push the Program Counter (PC) and vector to the first instruction of the interrupt service routine requires from 38 to 86 oscillator periods (2.38 to 5.38 μs @ 16 MHz). If the interrupt is due to an Immediate Command or DSC, additional time is required to read the Immediate Command or DSC SFR and vector to the appropriate service routine. This means two service routines back to back. One service routine to read the Slave Status (SSTAT) SFR to determine the source of the Request for Service interrupt, and second the service routine pointed to by the Immediate Command or DSC byte read from the respective SFR.

DMA

DMA is the fastest and most efficient way for the Host or internal CPU to communicate with the FIFO buffer. The UPI-452 provides support for both of these DMA paths. The two DMA paths and operations are fully independent of each other and can function simultaneously. While the Host DMA processor is performing a DMA transfer to or from the FIFO, the UPI-452 internal DMA processor can be doing the same.

Below are descriptions of both the Host and internal DMA operations. Both DMA paths can operate asynchronously and at different transfer rates. In order to make the most of this simultaneous asynchronous operation it is necessary to calculate the two transfer rates and accurately match their operations. Matching the different transfer rates is done by a combination of accurately programmed FIFO channel size and channel Thresholds. This provides the maximum Host and internal CPU to FIFO buffer interface decoupling. Below is a description of each of the two DMA operations and sample calculations for determining transfer rates. The next section of this application note, "Interface Latency", details the considerations involved in analyzing effective transfer rates when the overhead associated with each transfer is considered.

HOST FIFO DMA

DMA transfers between the Host and UPI-452 FIFO buffer are controlled by the Host CPU's DMA controller, and is independent of the UPI-452's internal two channel DMA processor. The UPI-452's internal DMA processor supports data transfers between the UPI-452 internal RAM, external RAM (via the Local Expansion Bus) and the various Special Function Registers including the FIFO Input and Output channel SFRs.

The maximum DMA transfer rate is achieved by the minimum DMA transfer cycle time to accomplish a source to destination move. The minimum Host UPI-452 FIFO DMA cycle time possible is determined by the $\overline{READ}$ and $\overline{WRITE}$ pulse widths, UPI-452 command recovery times in relation to the DMA transfer timing and DMA controller transfer mode used. Table 5 shows the relationship between the iAPX-286, iAPX-186 and UPI-452 for various DMA as well as non-DMA byte by byte transfer modes versus processor frequencies.

Host processor speed vs wait states required with UPI-452 running at 16 MHz:

Table 5. Host UPI-452 Data Transfer Performance

Processor & Speed	Wait States: Back to Back $\overline{READ}$/$\overline{WRITE}$'s	DMA: Single Cycle	Two Cycle
iAPX-186* 8 MHz	0	N/A*	0
10 MHz	0	N/A*	0
12.5 MHz	1	N/A*	0
iAPX-286** 6 MHz	0	0	0
8 MHz	1	1	0
10 MHz	2	2	0

NOTES:
* iAPX 186 On-chip DMA processor is two cycle operation only.
** iAPX 286 assumes 82258 ADMA (or other DMA) running 286 bus cycles at 286 clock rate.

In this application note system example, shown in Figure 1, DMA operation is assumed to be two bus cycle I/O to memory or memory to I/O. Two cycle DMA consists of a fetch bus cycle from the source and a store bus cycle to the destination. The data is stored in the DMA controller's registers before being sent to the destination. Single cycle DMA transfers involve a simultaneous fetch from the source and store to the destination. As the most common method of I/O-memory DMA operation, two cycle DMA transfers are the focus of this application note analysis. Equation 2 illustrates a calculation of the overall transfer rate between the FIFO buffer and external Host for a maximum FIFO size transfer. The equation does not account for the latency of initiating the DMA transfer.

Equation 2. Host FIFO DMA Transfer Rate—Input or Output Channel

2 Cycle DMA Transfer-I/O (UPI-452) to System Memory
= FIFO channel size* (DMA $\overline{READ}$/$\overline{WRITE}$ FIFO time + DMA $\overline{WRITE}$/$\overline{READ}$ Memory Time)
= 128 bytes* (200 ns + 200 ns)
= 51.2 μs
= 256 bus cycles*

NOTES:
*10 MHz iAPX 286, 200 ns bus cycles.

The UPI-452 design is optimized for high speed DMA transfers between the Host and the FIFO buffer. The

UPI-452 internal FIFO buffer control logic provides the necessary synchronization of the external Host event and the internal CPU machine cycle during UPI-452 $\overline{RD}/\overline{WR}$ accesses. This internal synchronization is addressed by the TCC AC specification of the UPI-452 shown in Figure 6. TCC is the time from the leading or trailing edge of a UPI-452 $\overline{RD}/\overline{WR}$ to the same edge of the next UPI-452 $\overline{RD}/\overline{WR}$. The TCC time is effectively another way of measuring the system bus cycle time with reference to UPI-452 accesses.

In the iAPX-286 10 MHz system depicted in this application note the bus cycle time is 200 ns. Alternate cycle accesses of the UPI-452 during two cycle DMA operation yields a TCC time of 400 ns which is more than the TCC minimum time of 375 ns. Back to back Host UPI-452 $\overline{READ}/\overline{WRITE}$ accesses may require wait states as shown in Table 5. The difference between 10 MHz iAPX-186 and 10 MHz iAPX 286 required wait states is due to the number of clock cycles in the respective bus cycle timings. The four clocks in a 10 MHz iAPX 186 bus cycle means a minimum TCC time of 400 ns versus 200 ns for a 10 MHz iAPX 286 with two clock cycle zero wait state bus cycle.

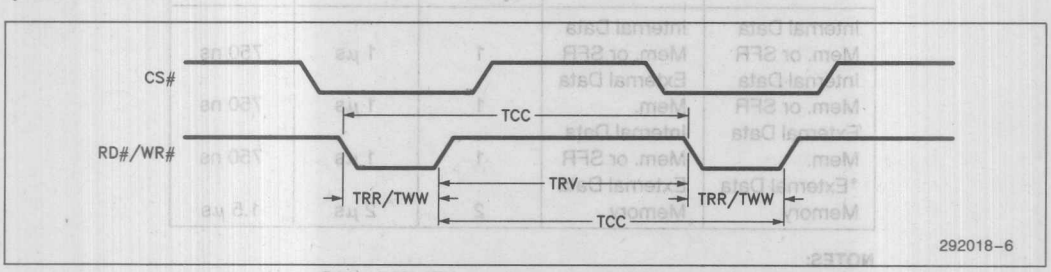
DMA handshaking between the Host DMA controller and the UPI-452 is supported by three pins on the UPI-452; DRQIN/INTRQIN, DRQOUT/INTRQOUT and $\overline{DACK}$. The DRQIN/INTRQIN and DRQOUT/INTRQOUT outputs are two multiplexed DMA or interrupt request pins. The function of these pins is controlled by MODE SFR bit 6 (MD6). DRQIN and DRQOUT provide a direct interface to the Host system DMA controller (see Figure 1). In response to a DRQIN or DRQOUT request, the Host DMA controller initiates control of the system bus using HLD/HLDA. The FIFO Input or Output channel transfer is accomplished with a minimum of Host overhead and system bus bandwidth.

The third handshake signal pin is $\overline{DACK}$ which is used as a chip select during DMA data transfers. The UPI-452 Host $\overline{READ}$ and $\overline{WRITE}$ input signals select the respective Input and Output FIFO channel during DMA transfers. The $\overline{CS}$ and address lines provide DMA acknowledge for processors with onboard DMA controllers which do not generate a $\overline{DACK}$ signal.

The iAPX 286 Block I/O Instructions provide an alternative to two cycle DMA data transfers with approximately the same data rate. The String Input and Output instructions (INS & OUTS) when combined with the Repeat (REP) prefix, modifies INS and OUTS to provide a means of transferring blocks of data between I/O and Memory. The data transfer rate using REP INS/OUTS instructions is calculated in the same way as two cycle DMA transfer times. Each $\overline{READ}$ or $\overline{WRITE}$ would be 200 ns in a 10 MHz iAPX 286 system. The maximum transfer rate possible is 2.5 MBytes/second. The Block I/O FIFO data transfer calculation is the same as that shown in Equation 2 for two cycle DMA data transfers including TCC timing effects.

FIFO Data Structure and Host DMA

During a Host DMA write to the FIFO, if a DSC is to be written, the DMA transfer is stopped, the DSC is written and the DMA restarted. During a Host DMA read from the FIFO, if a DSC is loaded into the I/O Buffer Latch the DMA request, DRQOUT, will be deactivated (see Figure 2 above). The Host Status (HSTAT) SFR Data Stream Command bit is set and the INTRQ interrupt output goes active, if enabled. The Host responds to the interrupt as described above.



Symbol	Description	Var. Osc.	@16 MHz
TCC	Command Cycle Time	6 * TcIcl	375 ns min
TRV	Command Recovery Time	75	75 ns min

Figure 6. UPI-452 Command Cycle Timing

Once INTRQ is deactivated and the DSC has been read by the Host, the DMA request, $\overline{\text{DRQOUT}}$, is reasserted by the UPI-452. The DMA request then remains active until the transfer is complete or another DSC is loaded into the I/O Buffer Latch.

An Immediate Command written by the internal CPU during a Host DMA FIFO transfer also causes the Host Status flag and INTRQ to go active if enabled. In this case the Immediate Command would not terminate the DMA transfer unless terminated by the Host. The INTRQ line remains active until the Host reads the Host Status (HSTAT) SFR to determine the source of the interrupt.

The net effect of a Data Stream Command (DSC) on DMA data transfer rates is to add an additional factor to the data transfer rate equation. This added factor is shown in Equation 3. An Immediate Command has the same effect on the data transfer rate if the Immediate Command interrupt is recognized by the Host during a DMA transfer. If the DMA transfer is completed before the Immediate Command interrupt is recognized, the effect on the DMA transfer rate depends on whether the block being transmitted is larger than the FIFO channel size. If the block is larger than the programmed FIFO channel size the transfer rate depends on whether the Immediate Command flag or interrupt is recognized between partial block transfers.

The FIFO configuration shown in Equation 3 is arbitrary since there is no way of predicting the size relative to when a DSC would be loaded into the I/O Buffer Latch. The Host DMA rate shown is for a UPI-452 (Memory Mapped or I/O) to 286 System Memory transfer as described earlier. The equations do not account for the latency of intiating the DMA transfer.

Equation 3. Minimum host FIFO DMA Transfer Rate Including Data Stream Command(s)

Minimum Host/FIFO DMA Transfer Rate w/ DSC
= FIFO size* Host DMA 2 cycle time transfer rate + iAPX 286 interrupt response time (Eq. #1)
= (32 bytes* (200 ns + 200 ns)) + 2.3 μs
= 15.1 μs
= 75.5 bus cycles (@10 MHz iAPX286, 200 ns bus cycle)

UPI-452 INTERNAL DMA PROCESSOR

The two identical internal DMA channels allow high speed data transfers from one UPI-452 writable memory space to another. The following UPI-452 memory spaces can be used with internal DMA channels:

Internal Data Memory (RAM)
External Data Memory (RAM)
Special Function Registers (SFR)

The FIFO can be accessed during internal DMA operations by specifying the FIFO IN (FIN) SFR as the DMA Source Address (SAR) or the FIFO OUT (FOUT) SFR as the Destination Address (DAR). Table 6 lists the four types of internal DMA transfers and their respective timings.

Table 6. UPI-452 Internal DMA Controller Cycle Timings

Source	Destination	Machine Cycles**	@12 MHz	@16 MHz
Internal Data Mem. or SFR	Internal Data Mem. or SFR	1	1 μs	750 ns
Internal Data Mem. or SFR	External Data Mem.	1	1 μs	750 ns
External Data Mem.	Internal Data Mem. or SFR	1	1 μs	750 ns
*External Data Memory	External Data Memory	2	2 μs	1.5 μs

NOTES:
*External Data Memory DMA transfer applies to UPI-452 Local Bus only.
**MSC-51 Machine cycle = 12 clock cycles (TCLCL).

FIFO Data Structure and Internal DMA

The effect of Data Stream Commands and Immediate Commands on the internal DMA transfers is essentially the same as the effect on Host FIFO DMA transfers. Recognition also depends upon the programmed DMA Mode, the interrupts enabled, and their priorities. The net internal effect is the same for each possible internal case. The time required to respond to the Immediate or Data Stream Command is a function of the instruction time required. This must be calculated by the user based on the instruction cycle time given in the MSC-51 Instruction Set description in the Intel Microcontroller Handbook.

It is important to note that the internal DMA processor modes and the internal FIFO logic work together to automatically manage internal DMA transfers as data moves into and out of the FIFO. The two most appropriate internal DMA processor modes for the FIFO are FIFO Demand Mode and FIFO Alternate Cycle Mode. In FIFO Demand Mode, once the correct Slave Control and DMA Mode bits are set, the internal Input FIFO channel DMA transfer occurs whenever the Slave Control Input FIFO Request for Service flag is set. The DMA transfer continues until the flag is cleared or when the Input FIFO Read Pointer SFR (IRPR) equals zero. If data continues to be entered by the Host, the internal DMA continues until an internal interrupt of higher priority, if enabled, interrupts the DMA transfer, the internal DMA byte count reaches zero or until the Input FIFO Read Pointer equals zero. A complete description of interrupts and DMA Modes can be found in the UPI-452 Data Sheet.

DMA Modes

The internal DMA processor has four modes of operation. Each DMA channel is software programmable to operate in either Block Mode or Demand Mode. Demand Mode may be further programmed to operate in Burst or Alternate Cycle Mode. Burst Mode causes the internal processor to halt its execution and dedicate its resources exclusively to the DMA transfer. Alternate Cycle Mode causes DMA cycles and instruction cycles to occur alternately. A detailed description of each DMA Mode can be found in the UPI-452 Data Sheet.

INTERFACE LATENCY

The interface latency is the time required to accommodate all of the overhead associated with an individual data transfer. Data transfer rates between the Host system and UPI-452 FIFO, with a block size less than or equal to the programmed FIFO channel size, are calculated using the Host system DMA rate. (see Host DMA description above). In this case, the entire block could be transferred in one operation. The total latency is the time required to accomplish the block DMA transfer, the interrupt response or poll of the Host Status SFR response time, and the time required to initate the Host DMA processor.

A DMA transfer between the Host and UPI-452 FIFO with a block size greater than the programmed FIFO channel size introduces additional overhead. This additional overhead is from three sources; first, is the time to actually perform the DMA transfer. Second, the overhead of initializing the DMA processor, third, the handshaking between each FIFO block required to transfer the entire data block. This could be time to wait for the FIFO to be emptied and/or the interrupt response time to restart the DMA transfer of the next portion of the block. A fourth component may also be the time required to respond to Underrun and Overrun FIFO Errors.

Table 7 shows six typical FIFO Input/Output channel sizes and the Host DMA transfers times for each. The timings shown reflect a 10 MHz system bus two cycle I/O to Memory DMA transfer rate of 2.5 MBytes/second as shown in Equation 1. The times given would be the same for iAPX 286 I/O block move instructions REP INS and REP OUTS as described earlier.

Table 7. Host DMA FIFO Data Transfer Times

FIFO Size:	32	43	64	85	96	128	bytes
Full or Empty	1/4	1/3	1/2	2/3	3/4	Full or Empty	
Time	12.8	17.2	25.6	34.0	38.4	51.2	μs

Table 8 shows six typical FIFO Input/Output channel sizes and the internal DMA processor data transfers times for each. The timings shown are for a UPI-452 single cycle Burst Mode transfer at 16 MHz or 750 ns per machine cycle in or out of the FIFO channels. The

machine cycle time is that of the MSC-51 CPU; 6 states, 2 XTAL2 clock cycles each or 12 clock cycles per machine cycle. Details on the MSC-51 machine cycle timings and operation may be found in the Intel Microcontroller Handbook.

Table 8. UPI-452 Internal DMA FIFO Data Transfer Times

FIFO Size:	32	43	64	85	96	128	bytes
Full or Empty	1/4	1/3	1/2	2/3	3/4	Full or Empty	
Time	24.0	32.3	48.0	64.6	72.0	96.0	μs

A larger than programmed FIFO channel size data block internal DMA transfer requires internal arbitration. The UPI-452 provides a variety of features which support arbitration between the two internal DMA channels and the FIFO. An example is the internal DMA processor FIFO Demand Mode described above. FIFO Demand Mode DMA transfers occur continuously until the Slave Status Request for Service Flag is deactivated. Demand Mode is especially useful for continuous data transfers requiring immediate attention. FIFO Alternate Cycle Mode provides for interleaving DMA transfers and instruction cycles to achieve a maximum of software flexibility. Both internal DMA channels can be used simultaneously to provide continuous transfer for both Input and Output FIFO channels.

Byte by byte transfers between the FIFO and internal CPU timing is a function of the specific instruction cycle time. Of the 111 MCS-51 instructions, 64 require 12 clock cycles, 45 require 24 clock cycles and 2 require 48 clock cycles. Most instructions involving SFRs are 24 clock cycles except accumulator (for example, MOV direct, A) or logical operations (ANL direct, A). Typical instruction and their timings are shown in Table 9.

Oscillator Period: @ 12 MHz = 83.3 ns
@ 16 MHz = 62.5 ns

Table 9. Typical Instruction Cycle Timings

Instruction	Oscillator Periods	@12 MHz	@16 MHz
MOV direct†, A	12	1 μs	750 ns
MOV direct, direct	24	2 μs	1.5 μs

NOTE:
† Direct = 8-bit internal data locations address. This could be an Internal Data RAM location (0–255) or a SFR [i.e., I/O port, control register, etc.]

Byte by byte FIFO data transfers introduce an additional overhead factor not found in internal DMA operations. This factor is the FIFO block size to be transferred; the number of empty locations in the Output channel, or the number of bytes in the Input FIFO channel. As described above in the FIFO Data Structure section, the block size would have to be determined by reading the channel read and write pointer and calculating the space available. Another alternative uses the FIFO Overrun and Underrun Error flags to manage the transfers by accepting error flags. In either case the instructions needed have a significant impact on the internal FIFO data transfer rate latency equation.

A typical effective internal FIFO channel transfer rate using internal DMA is shown in Equation 4. Equation 5 shows the latency using byte by byte transfers with an arbitrary factor added for internal CPU block size calculation. These two equations contrast the effective transfer rates when using internal DMA versus individual instructions to transfer 128 bytes. The effective transfer rate is approximately four times as fast using DMA versus using individual instructions (96 μs with DMA versus 492 μs non-DMA).

Equation 4. Effective Internal FIFO Transfer Time Using Internal DMA

Effective Internal FIFO Transfer Rate with DMA
= FIFO channel size * Internal DMA Burst Mode Single Cycle DMA Time
= 128 Bytes * 750 ns
= 96 μs

Equation 5. Effective FIFO Transfer Time Using Individual Instructions

Effective Internal FIFO Transfer Rate without DMA
= FIFO channel size * Instruction Cycle Time + Block size calculation Time
= 128 Bytes * (24 oscillator periods @ 16 MHz) + 20 instructions (24 oscillator period each @ 16 MHz)
= 128 * 1.5 μs + 300 μs
= 492 μs

FIFO DMA FREEZE MODE INTERFACE

FIFO DMA Freeze Mode provides a means of locking the Host out of the FIFO Input and Output channels. FIFO DMA Freeze Mode can be invoked for a variety of reasons, for example, to reconfigure the UPI-452 Local Expansion Bus, or change the baud rate on the serial channel. The primary reason the FIFO DMA Freeze Mode is provided is to ensure that the Host does not read from or write to the FIFO while the FIFO interface is being altered. ONLY the internal CPU has the capability of altering the FIFO Special Function Registers, and these SFRs can ONLY be altered during FIFO DMA Freeze Mode. FIFO DMA Freeze Mode inhibits Host access of the FIFO while the internal CPU reconfigures the FIFO.

FIFO DMA Freeze Mode should not be arbitrarily invoked while the UPI-452 is in normal operation. Because the external CPU runs asynchronously to the internal CPU, invoking freeze mode without first properly resolving the FIFO Host interface may have serious consequences. Freeze Mode may be invoked only by the internal CPU.

The internal CPU invokes Freeze Mode by setting bit 3 of the Slave Control SFR (SC3). This automatically forces the Slave and Host Status SFR FIFO DMA Freeze Mode to In Progress (SSTAT SST5 = 0, HSTAT SFR HST1 = 1). INTRQ goes active, if enabled by MODE SFR bit 4, whenever FIFO DMA Freeze Mode is invoked to notify the Host. The Host reads the Host Status SFR to determine the source of the interrupt. INTRQ and the Slave and Host Status FIFO DMA Freeze Mode bits are reset by the Host $\overline{\text{READ}}$ of the Host Status SFR.

During FIFO DMA Freeze Mode the Host has access to the Host Status and Control SFRs. All other Host FIFO interface access is inhibited. Table 10 lists the FIFO DMA Freeze Mode status of all slave bus interface Special Function Registers. The internal DMA processor is disabled during FIFO DMA Freeze Mode and the internal CPU has write access to all of the FIFO control SFRs (Table 11).

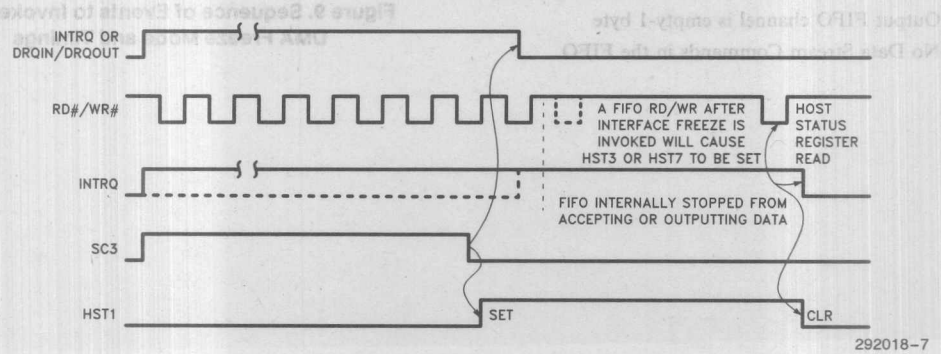
If FIFO DMA Freeze Mode is invoked without stopping the host, only the last two bytes of data written into or read from the FIFO will be valid. The timing diagram for disabling the FIFO module to the external Host interface is illustrated in Figure 7. Due to this synchronization sequence, the UPI-452 might not go into FIFO DMA Freeze Mode immediately after the Slave Control SFR FIFO 7 DMA Freeze Mode bit (SC3) is set = 0. A special bit in the Slave Status SFR (SST5) is provided to indicate the status of the FIFO DMA Freeze Mode. The FIFO DMA Freeze Mode operations described in this section are only valid after SST5 is cleared.

Either the Host or internal CPU can request FIFO DMA Freeze Mode. The first step is to issue an Immediate Command indicating that FIFO DMA Freeze Mode will be invoked. Upon receiving the Immediate Command, the external CPU should complete servicing all pending interrupts and DMA requests, then send an Immediate Command back to the internal CPU acknowledging the FIFO DMA Freeze Mode request. After issuing the first Immediate Command, the internal CPU should not perform any action on the FIFO until FIFO DMA Freeze Mode is invoked. The handshaking is the same in reverse if the HOST CPU initiates FIFO DMA Freeze Mode.

After the slave bus interface is frozen, the internal CPU can perform the operations listed below on the FIFO Special Function Registers. These operations are allowed only during FIFO DMA Freeze Mode. Table 11 summarizes the characteristics of all the FIFO Special Function Registers during Normal and FIFO DMA Freeze Modes.

For FIFO Reconfiguration
1. Changing the Channel Boundary Pointer SFR.
2. Changing the Input and Output Threshold SFR.

To Enhance the testability
3. Writing to the read and write pointers of the Input and Output FIFO's.
4. Writing to and reading the Host Control SFRs.
5. Controlling some bits of Host and Slave Status SFRs.
6. Reading the Immediate Command Out SFR and Writing to the Immediate Command in SFR.



NOTE:
Timing Diagram of disabling of FIFO Module—External Host Interface.

Figure 7. Disabling FIFO to Host Slave Interface Timing Diagram

The sequence of events for invoking FIFO DMA Freeze Mode are listed in Figure 8.

1. Immediate Command to request FIFO DMA Freeze Mode (interrupt)
2. Host/internal CPU interrupt response/service
3. Host/internal CPU clear/service all pending interrupts and FIFO data
4. Internal CPU sets Slave Control (SLCON) FIFO DMA

   Freeze Mode bit = 0, FIFO DMA Freeze Mode, Host Status SFR FIFO DMA Freeze Mode Status bit = 1, INTRQ active (high)
5. Host $\overline{\text{READ}}$ Host Status SFR
6. Internal CPU reconfigures FIFO SFRs
7. Internal CPU resets Slave Control (SLCON) FIFO DMA

   Freeze Mode bit = 1, Normal Mode, Host Status FIFO DMA Freeze Mode Status bit = 0.
8. Internal CPU issues Immediate Command to Host indicating that FIFO DMA Freeze Mode is complete

   or

   Host polls Host Status SFR FIFO DMA Freeze Mode bit to determine end of reconfiguration

Figure 8. Sequence of Events to Invoke FIFO DMA Freeze Mode

EXAMPLE CONFIGURATION

An example of the time required to reconfigure the FIFO 180 degrees, for example from 128 bytes Input to 128 bytes Output, is shown in Figure 9. The example approximates the time based on several assumptions;

1. The FIFO Input channel is full-128 bytes of data
2. Output FIFO channel is empty-1 byte
3. No Data Stream Commands in the FIFO.
4. The Immediate Command interrupt is responded to immediately—highest priority—by Host and internal CPU.
5. Respective interrupt response times
   a. Host (Equation 3 above) = approximately 1.6 μs
   b. Internal CPU is 86 oscillator periods or approximately 5.38 μs worst case.

Event	Time
Immediate Command from Host to UPI-452 to request FIFO DMA Freeze Mode (iAPX286 $\overline{\text{WRITE}}$)	0.30 μs
Internal CPU interrupt response/service	5.38 μs
Internal CPU clears FIFO-128 bytes DMA	96.00 μs
Internal CPU sets Slave Control Freeze Mode bit	0.75 μs
Immediate Command to Host-Freeze Mode in progress Host Immediate Command interrupt response	2.3 μs
Internal CPU reconfigures FIFO SFRs	
Channel Boundary Pointer SFR	0.75 μs
Input Threshold SFR	0.75 μs
Output Threshold SFR	0.75 μs
Internal CPU resets Slave Control (SLCON) Freeze Mode bit = 1, Normal Mode, and automatically resets Host Status FIFO DMA Freeze Mode bit	2.3 μs
Internal CPU writes Immediate Command Out	0.75 μs
Host Immediate Command interrupt service	2.3 μs
Total Minimum Time to Reconfigure FIFO	112.33 μs

Figure 9. Sequence of Events to Invoke FIFO DMA Freeze Mode and Timings

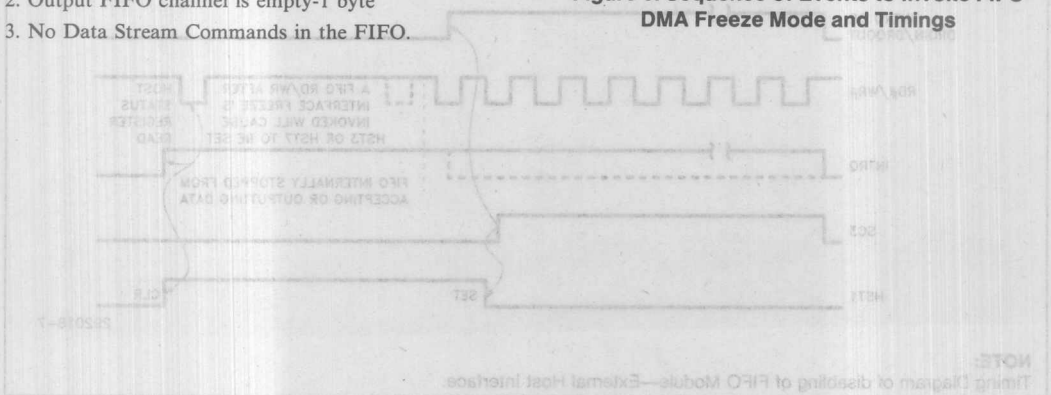


Table 10. Slave Bus Interface Status During FIFO DMA Freezer Mode

Interface Pins;							Operation In Normal Mode	Status In Freeze Mode
$\overline{DACK}$	$\overline{CS}$	A2	A1	A0	$\overline{READ}$	$\overline{WRITE}$		
1	0	0	1	0	0	1	Read Host Status SFR	Operational
1	0	0	1	1	0	1	Read Host Control SFR	Operational
1	0	0	1	1	1	0	Write Host Control SFR	Disabled
1	0	0	0	0	0	1	Data or DMA data from Output Channel	Disabled
1	0	0	0	0	1	0	Data or DMA data to Input Channel	Disabled
1	0	0	0	1	0	1	Data Stream Command from Output Channel	Disabled
1	0	0	0	1	1	0	Data Stream Command to Input Channel	Disabled
1	0	1	0	0	0	1	Read Immediate Command Out from Output Channel	Disabled
1	0	1	0	0	1	0	Write Immediate Command In to Input Channel	Disabled
0	X	X	X	X	0	1	DMA Data from Output Channel	Disabled
0	X	X	X	X	1	0	DMA Data to Input Channel	Disabled

NOTE:
X = don't care

Table 11. FIFO SFR's Characteristics During FIFO DMA Freeze Mode

Label	Name	Normal Operation (SST5 = 1)	Freeze Mode Operation (SST5 = 0)
HCON	Host Control	Not Accessible	Read & Write
HSTAT	Host Status	Read Only	Read & Write
SLCON	Slave Control	Read & Write	Read & Write
SSTAT	Slave Status	Read Only	Read & Write
IEP	Interrupt Enable & Priority	Read & Write	Read & Write
MODE	Mode Register	Read & Write	Read & Write
IWPR	Input FIFO Write Pointer	Read Only	Read & Write
IRPR	Input FIFO Read Pointer	Read Only	Read & Write
OWPR	Output FIFO Write Pointer	Read Only	Read & Write
ORPR	Output FIFO Read Pointer	Read Only	Read & Write
CBP	Channel Boundary Pointer	Read Only	Read & Write
IMIN	Immediate Command In	Read Only	Read & Write
IMONT	Immediate Command Out	Read & Write	Read & Write
FIN	FIFO IN	Read Only	Read Only
CIN	COMMAND IN	Read Only	Read Only
FOUT	FIFO OUT	Read & Write	Read & Write
COUT	COMMAND OUT	Read & Write	Read & Write
ITHR	Input FIFO Threshold	Read Only	Read & Write
OTHR	Other FIFO Threshold	Read Only	Read & Write

intel

ARTICLE REPRINT

AR-409

October 1985

Increased Functions In Chip Result In Lighter, Less Costly Portable Computer

Jafar Modares,
Application Engineer,
Intel Corporation
Chandler, Arizona

© INTEL CORPORATION, 1985

ORDER NUMBER: 270126-001

Reprinted from **Design News, 8-19-85**

270126-1

INCREASED FUNCTIONS IN CHIP RESULT IN LIGHTER, LESS COSTLY PORTABLE COMPUTER

Jafar Modares, Applications Engineer, Intel Corp., Chandler, AZ

Advances in technology have made it possible to reduce the size and increase the functionality and performance of computers and computer peripherals. With the help of microtechnology it is possible to construct a computer terminal that is smaller and lighter than a briefcase, and that can be connected to a mainframe from almost anywhere.

As more portable computers are introduced to the marketplace, the demand for lighter and even smaller systems at a lower cost are inevitable. To meet this demand changes in the architecture are necessary.

With Intel's recently introduced 80C51BH microcontroller several obstacles in the design of the portable computer have been overcome. The 80C51BH is a single chip 8-bit microcontroller that requires a single 5V power supply. It has 32 I/O lines. Its functionalities include excellent bit and byte manipulation capability at extremely high speeds as well as interfacing flexibilities through the serial and parallel channels to intelligent and unintelligent devices. It can carry its own program memory up to 4 Kbytes and has various tools and support systems.

This article discusses the implementation of a prototype portable terminal based on Intel's new 80C51BH microcontroller, and introduces the tools and techniques available to build such a computer terminal. In the application discussed, the chip monitors the keyboard, communicates with the host computer at very high BAUD rates, and displays information on the screen at slower rates for human beings. The chip also monitors the power supply for switching to the battery in case of power failure to save valuable data and computer time. The prototype is currently under futher development at Intel's Microcontroller Division.

Introducing the 80C51BH

Very new in the market, the 80C51BH is a member of Intel's MCS-51 family. The MCS-51 is a group of 8-bit microcontrollers that are extremely powerful because of their I/O structure and their bit manipulating capabilities. The 80C51BH has 4 Kbytes of on-chip program memory with the capability to address another 60 Kbytes of external program memory. In addition it has 128 bytes of on-chip RAM and can access 64 Kbytes of external data memory.

Since the 80C51BH is CMOS, it has very low power consumption (15 mA at 5V, 12 MHz). In addition, it has two power saving features not available in HMOS versions of the family. These are the Idle and Power Down modes, which are controlled by software and further reduce power comsumption. These power saving features make it ideal for battery-operated backed-up systems.

Display device

The display device of this portable terminal is a 25-line × 80-character Liquid Crystal Display (LCD). It is capable of displaying the same number of characters as a typical CRT, and it is not as bulky or heavy as the latter.

LCDs can be divided into two large categories: smart LCDs and dumb LCDs. Those displays that have the capability to receive a byte of ASCII and translate it into a displayed character are considered smart. On the other hand, dumb displays are only a matrix of dots. The former type has some kind of controller of its own plus a small memory to hold the look-up table of characters (character generator). When using a dumb display the microcontroller has to address and turn on the correct dots to make a character, this means more I/O pins will be required. However, it gives extra capabilities such as graphic displays and special or custom character generation.

Both types of displays normally accept data through an 8-bit bus. Although the LCD is relatively slow, it can still share the bus with a memory device without any degradation of the system performance.

Keyboard

Depending on their task and purpose, keyboards vary in shape, size, and the number of keypads. The keyboard for a computer terminal, as a minimum, should have all the alphanumeric keys (standard typewriter) plus a Control, an Escape, and optionally, some Function keys.

As the diagram in Figure 1 shows a

270126-2

typical keyboard consists of a matrix of eight scan lines and eight receive lines. The scan lines are connected to Port 0 of the microcontroller, and receive lines are connected to Port 1. The software writes 0s to Port 0 to hold the scan lines at a logic Low, and it writes 1s to Port 1 to hold the receive lines at a logic High. Pressing one of the keys connects a scan line to receive a line and pulls that receive line Low.

Besides being used for the scan lines, Port 0 is also the bus for the data buffer RAM and the display unit. While the controller is talking to the RAM or to the display, the bus must not be used by other devices or bus contention can occur. Since the microcontoller initiates the access to the display and to the data buffer RAM, no conflict can occur between them. However, if more than one key on the same receive line is held down simultaneously while the RAM or the display is being accessed, it is possible to foul up the information being transferred. Thus, to avoid bus contention, diodes D1 through D8 are placed on the scan lines, as shown in Figure 1.

All the receive lines are ANDed to External Interrupt 1, so that pulling any of the receive lines Low will generate an interrupt. The interrupt service routine, adapts the "two key lock-out" system and identifies the pressed key. This system allows only one key to be pressed simultaneously, all of them will be ignored.

On some keyboards, certain keys (such as Shift, Control or Escape) are not a part of the line matrix. These keys connect directly to a port pin on the microcontroller. They would not cause lock-out if pressed simultaneously with a matrix key, nor generate an interrupt if pressed singly. However, if they are part of the matrix, then the software has to recognize those keys and take proper action depending on the function of the pressed key.

Normally, when a key is pressed, the microcontroller is in the Idle mode and no other task is in progress. Pressing a key on the matrix generates an interrupt, which terminates the Idle mode. The interrupt service routine first calls a subroutine to provide a delay of approximately 25 msec (to debounce the key), and to perform the task of identifying which key is down.

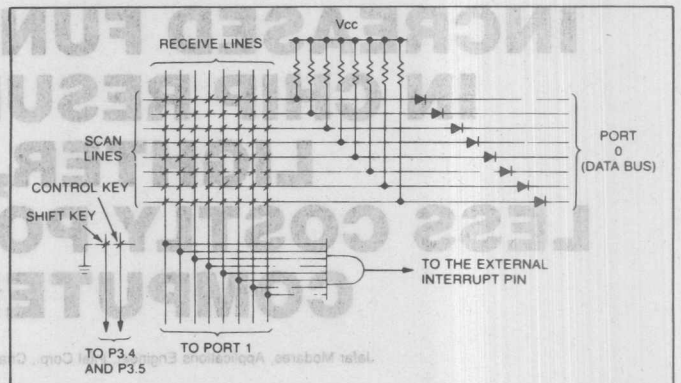


Figure 1

There are a number of ways that the interrupt service routine can identify the press key. One way is to utilize the bit-addressing capabilities of the 80C51BH to test each bit of the receive lines for a zero, which generated the interrupt, and records the bit position. Then write 0s to the receive lines, 1s to the scan lines, test the scan lines for a zero, and record its position. If the controller finds more than one zero on each port, it will discontinue any further processing and return to the main program.

Once the bit positions that contain 0 are recorded, they are used as the address for the characters in the look-up table. The subroutine finds the character that corresponds to the pressed key. The character is represented in ASCII code.

The look-up table is a list of the ASCII representation of each keypad character, and is stored in the program memory. The order in which they sit corresponds to the address generated by the scan subroutine when that character is pressed on the keyboard.

Serial communication

Once the 80C51BH has the ASCII code generated from a key closure, it can send it through its Serial Channel to the host computer. The serial port of the microcontroller can be programmed for all of the standard rates up to 375K. If the terminal is to be connected directly to the mainframe, a simple circuit translates the TTL levels to the RS232 level. The circuit also eliminates the need for the −12V supply required for RD232. The circuit diagram is shown in Figure 2.

However, the primary application of this terminal is to be the traveling person's window to the central system from any remote location. In this application a modem is needed. There are many types of modems available. Some are on PC boards for OEM use and others are ready to connect directly to the telephone by the user. They also vary in size, performance, and the way they communicate. A proper modem for the portable terminal should be small enough to carry in a briefcase and preferably be a low-power device. When an ASCII code is received by the host computer, it records that code and echoes it back to the microcontroller which displays it on the LCD.

The serial port of the 80C51BH can generate interrupts every time it receives or transmits information. The serial port interrupt must have service priority over the external interrupt generated by the keyboard. The high priority enables the serial port to receive data any time the computer addresses the terminal and transmits data.

The serial port interrupt service routine transfers the received data to the display or to a memory buffer, depend-

270126–3

ing on the mode selected by a function key. One mode of this function key tells the controller to move the received data directly to the display. The other mode stores pages of the information in the buffer RAM, so that the user can edit or display the data, one page at a time, on the LCD without using the main system's CPU time.

Data buffer memory

The data buffer RAM is temporary storage for the information which is either generated at the terminal or retrieved from the central system. The user can display portions of the stored information or scroll through it. The user can also alter the data on the terminal and transmit it back to the computer in a block form.

A suitable device for this purpose is the 51C86 iRAM. This device is a pseudo-static dynamic RAM that has a built-in address latch. An internal high-speed arbitration circuit resolves any potential conflict arising between read/write and internal refresh cycles.

This iRAM is 8K × 8-bit and is suf-

routine takes the data from the SBUF register and writes it to the iRAM. Two index pointers in the serial port interrupt service routine help keep track of the incoming and outgoing iRAM data.

There are times the controller has to wait for reasons such as the debounce delay in the scan subroutine or when writing to the display, and must slow down. While the 80C51BH is waiting, it goes into the Idle mode until one of the timers, which programmed to run, times out and generates an interrupt. The interrupt service routine for this type of interrupt is a short one, it sets or clears one or more flags and returns to continue the task that was in progress before going into the Idle mode.

The controller is also in the Idle mode when there is no activity in the terminal, i.e., no data is being received or transmitted, and the keyboard is not being used. Therefore, is is appropriate to say that when power is applied to the terminal, the controller spends more than 90% of its time in the Idle mode.

What is the Idle mode?

When Bit 0 in the Special Function Register PCON is set, the CPU gates off

Since the CPU consumes about 90% of the chip's power, this process saves a significant amount of power. More importantly the on chip peripherals and the RAM continue their normal functions independently of the CPU. Since the oscillator is still running, any enabled interrupt (internal or external) will wake the CPU up from its sleep, and it will start executing instructions from the interrupt service routine.

A true portable terminal should be capable of operating from a battery source. There are occasions when one would like to use the terminal at a location where an electric outlet is not readily available. But the main purpose of the battery supply is to save the data, which has been entered and stored in the buffer RAM, in the case of an unexpected power failure.

While performing all of the previously mentioned tasks, the 80C51BH can monitor its power supply, detect a power loss in its earliest stages, and initiate a power Down to save the data of the internal and external RAM.

One method for the 80C51BH to monitor its power supply is to have the positive half-cycles of the power supply transformer fed to the External Interrupt 0 pin (Figure 3). In the level activated mode, this pin generates an interrupt every time there is a high to low transition. The interrupt service routine reloads Timer 0 to a value that will make it overflow sometime between one and two periods of the line frequency. As long as the half cycles keep coming in, the timer never overflows, because it is reloaded every half a cycle. If there is a single half a cycle in which the line voltage does not reach a high enough level to generate the interrupt, the timer rolls over and generates a timer interrupt.

The interrupt service routine for Timer 0 saves the critical data of some of the internal registers and puts the controller into a Power down mode. A reset button restarts the microcontroller to resume operation when power is restored.

In this mode the CPU and all of the on-chip peripherals go to sleep. The device stops its oscillator, freezes all the

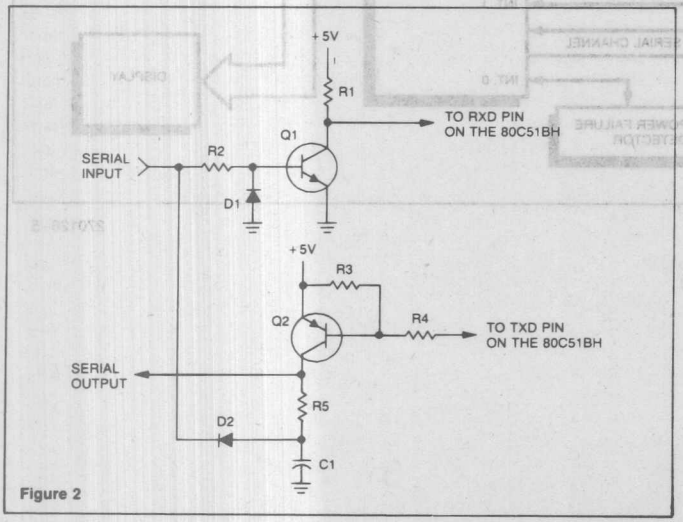


Figure 2

270126-4

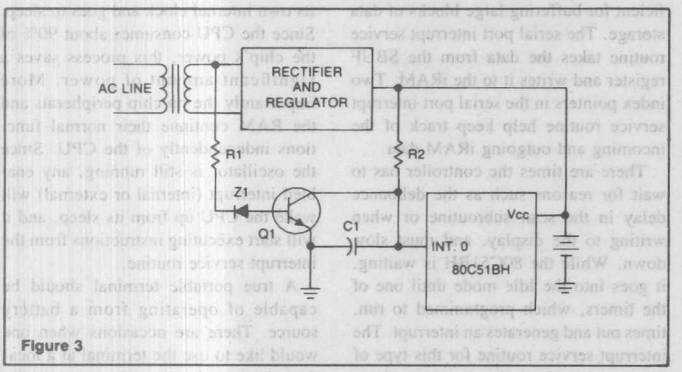


Figure 3

activities and saves the information in its internal RAM for as long as the supply voltage can be reduced to as low as 2 volts without running any risks of losing the internal RAM information. Supply current in this mode is normally 10 to 50 μA.

Bit 1 of the Special Function Register PCON controls this mode. The instruction that puts the device in the Power Down mode is the last one executed. The only way for the part to exit this mode is with a hardware reset.

A prototype of this terminal was built and connected to a MDS 800 developement system. The terminal communicated with the host computer through the serial channel at the rate of 2400 baud. The 80C51BH was emulated using Intel's ICE-51 in circuit emulator. The block diagram of the terminal is shown in Figure 4.

The CHMOS controller and LCD combination provides a system that requires only a single voltage power supply, and consumes less than 200 mW. The system's low power consumption eliminates the need for complex, voltage-regulating hardware, and cooling fans. The result is a lighter and smaller computer terminal at a very low cost. □

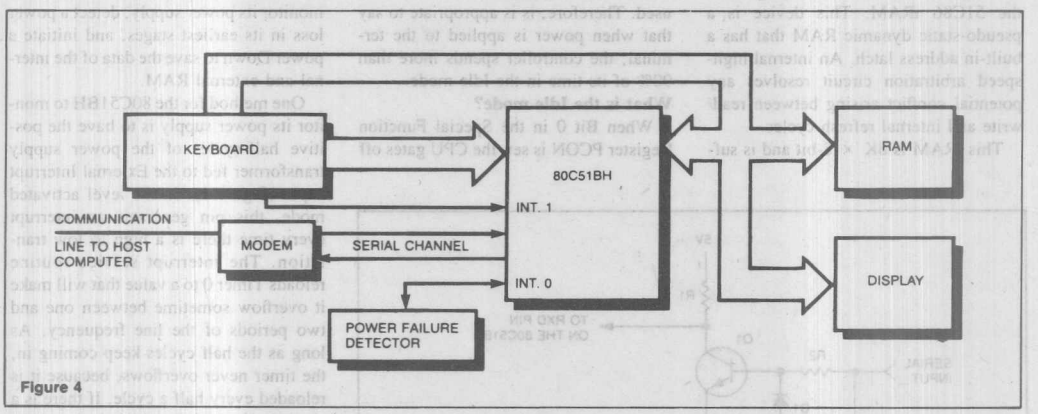


Figure 4

270126-5

IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. IE-32, NO. 4, NOVEMBER 1985

Using the 8051 Microcontroller with Resonant Transducers

TOM WILLIAMSON

Abstract—**Having to interface an analog transducer to a digital control system through an analog-to-digital converter represents an expensive bottleneck in the development of many systems. Some transducer companies are addressing this problem by developing proprietary families of resonant transducers.**

Resonant transducers are oscillators whose frequency depends in some known way on the physical property being measured. The electrical output from these devices is a train of rectangular pulses whose repetition rate encodes the value of the measurand. Changes in the measurand cause the frequency to shift. The microcontroller detects the frequency shift, runs a validity check on it, and converts it in software to the measurand value.

This paper discusses software interfacing techniques between resonant transducers and the 8051. Techniques for measuring frequency and period are discussed and compared for resolution and interrogation time. The 8051 is capable of performing these tasks in extremely short CPU time. Requirements for obtaining *n*-bit resolution in the measurement are discussed. It is determined that it is always faster to evaluate the measurand to a given level of resolution by measuring the period rather than the frequency, even if the measurand is proportional to the frequency rather than to the period. Numerical and software examples are presented to illustrate the concepts.

I. Resonant Transducers

MOST sensing transducers are not directly compatible with digital controllers, because they generate analog signals. A few transducer companies are developing proprietary families of sensors which generate signals that are more directly compatible with digital systems. These are not analog sensors with built-in A–D conversion, but oscillators whose frequency depends in some known way on the physical property being measured.

The technology is applicable to virtually any type of measurand: pressure, gas density, position, temperature, force, etc. The sensor and microcontroller can operate from the same supply voltage, so the sensor can in most cases connect directly to a port pin on the microcontroller.

The nominal reference frequency of the output signal from these devices is in the range of 20 Hz–500 kHz, depending on the design. A change in the measurand away from the reference condition causes the frequency to shift by an amount that is related to the change in the measurand value. Transducers are available that have a full-scale frequency shift of 2–1. The microcontroller detects the change in frequency or period and converts it in software to the measurand value.

II. Connecting the Digital Transducer to the 8051

Normally the transducer output can be connected directly to one of the 8051 port pins. An exception would occur when the transducer signal does not restrict itself to the voltage range of −0.5 to +5.5 V.

The 8051 is not sensitive to the rise and fall times of its input signals. It detects transitions by sampling its port pins at fixed intervals (once per machine cycle), and responds to a change in the sequence of samples. If the slew rate of the transducer signal is extremely slow, noise superimposed on the signal could cause the sequence of samples to show false transitions. There could on that account be situations in which the transducer signal should be buffered through a Schmitt Trigger to square it up.

III. Timer/Counter Structure in the 8051

The 8051 has two 16-bit timer/counters: Timer 0 and Timer 1. Both can be configured in software to operate either as timers or as event counters.

In the "timer" function, the register is automatically incremented every machine cycle. Since a machine cycle in the 8051 consists of 12 clock periods, the timer is being incremented at a constant rate of 1/12 the clock frequency.

In the "counter" function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin (*T*0 or *T*1). The way this function works is the external input pin is sampled once each machine cycle (once every 12 clock periods), and when the samples show a high in one cycle and a low in the next, the count is incremented.

Note too that since it takes two machine cycles (24 clock periods) to recognize a 1-to-0 transition, the maximum count rate is 1/24 the clock frequency. If the clock frequency is 12 MHz, the maximum count rate is 500 kHz. There are no requirements on the duty cycle of the signal being counted.

The 8052, an enhanced version of the 8051, has three 16-bit timer/counters, two of which are identical to those in the 8051. The third timer/counter can operate either as a 16-bit timer/counter with automatic reload to a preset 16-bit value on rollover, or as a 16-bit timer/counter with a "capture" mode. In the capture mode a 1-to-0 transition at the T2EX pin causes the current value in the counting register to the "captured" into RAM. The third timer makes the 8052 particularly easy to interface with resonant transducers.

IV. Whether to Measure Frequency or Period

Measuring the frequency requires counting transducer pulses for a fixed sample time. Measuring the period requires measuring elapsed time for a fixed number of transducer pulses. For a given level of accuracy in the determination of the value of the measurand, it is usually faster to measure the period, rather than the frequency, even if the measurand is

Manuscript received October 25, 1984.
The author is with Intel Corporation, Chandler, AZ 85224.

© 1985 IEEE

270434-1

proportional to frequency rather than period. However, both types of measurements will be discussed here.

Two timer/counters can be used, one to mark time and the other to count transducer pulses. If the frequency being counted does not exceed 50 kHz or so, one may equally well connect the transducer signal to an external interrupt pin, and count transducer pulses in software. That frees one timer, with very little cost in CPU time.

V. How to Measure Transducer Frequency

Measuring the frequency means counting transducer pulses for some desired sample time. The count that is directly obtained is $T \times F$, where T is the sample time and F is the frequency. The full scale range is $T \times (F\text{max} - F\text{min})$. For n-bit resolution

$$1\ \text{LSB} = \frac{T \times (F\text{max} - F\text{min})}{2^n}.$$

Therefore, the sample time required for n-bit resolution is

$$T = \frac{2^n}{F\text{max} - F\text{min}}.$$

For example, 8-bit resolution in the measurement of a frequency that varies between 5 and 10 kHz would require, according to this formula, a sample time of 51.2 ms. The maximum acceptable frequency count would be 51.2 ms × 10 kHz = 512 counts. The minimum would be 256 counts. Subtracting 256 from each frequency count would allow the frequency to be reported on a scale of 0 to *FF* in hex digits.

If *F*min and *F*max are closer together it takes more time to resolve them. 8-bit resolution in the measurement of a frequency that varies between 7 and 9 kHz would require a sample time of 128 ms. The maximum and minimum acceptable counts would be 1152 and 896. Subtracting 896 from each frequency count would allow the frequency to be reported on a scale of 0 to *FF* in hex digits.

To implement the measurement, one timer is used to establish the sample time. In this function it autoincrements every machine cycle. A machine cycle consists of 12 periods of the clock oscillator. The sample time can be converted to machine cycles by multiplying it by (*F*xtal)/12, where *F*xtal is the 8051 clock frequency. The timer needs to be preset so that it rolls over at the end of each sample time. Then it generates an interrupt, and the interrupt routine reads and clears the transducer pulse counter, and then reloads the timer with the correct preset value.

The preset or reload value is the two's complement negative of the sample time in machine cycles. For example, with a 12-MHz clock frequency, the reload value required to establish a 51.2 ms sample time is

$$-\frac{(51.2\ \text{ms}) \times (12000\ \text{kHz})}{12} = -51200 = 3800\ \text{H}.$$

In many cases the required sample time exceeds the capacity of a 16-bit timer. For example, establishing a 128 ms sample time with a 12-MHz clock frequency requires a 3-byte timer with a reload of FE0C00H. The 8051 timer, being only 2-bytes wide, can be augmented in software in the timer interrupt routine to three bytes. The 8051 has a DJNZ instruction (decrement and jump if not zero) which makes it easier to code the third timer byte to count down instead of up. If the third timer byte counts down, its reload value is the two's complement of what it would be for an up-counter. For example, if the two's complement of the sample time is FE0C00H, then the reload value for the third timer byte would be 02, instead of *FE*. The timer interrupt routine might then be

```
    DJNZ    THIRD_TIMER_BYTE,OUT
    MOV     TL0,#0
    MOV     TH0,#0CH
    MOV     THIRD_TIMER_BYTE,#02
(Now read and clear the
 transducer pulse counter.)
OUT: RETI
```

Interrupt latency will have no effect on the measurement if the latency is the same for every sample time.

The trouble with measuring the frequency is it is not only slow, but a waste of the resolving power of the 8051's timers. A timer with microsecond resolution is being used to mark off 100-ms time periods. The technique is nevertheless useful if the timer is already serving other purposes (servicing a display, perhaps), so that the sample time is coming relatively free of charge. But in most cases it is faster and equally accurate to measure the frequency by deriving it from a measurement of the period.

VI. How to Measure the Period

Measuring the period of the transducer signal means measuring the total elapsed time over N-transducer pulses. The quantity that is directly measured is $N \times T$, where T is the period of the transducer signal in *machine cycles*. The relationship between T in machine cycles and the transducer frequency F in arbitrary frequency units is

$$T = \frac{F\text{xtal}}{F} \times (1/12)$$

where *F*xtal is the 8051 clock frequency, in the same unit as F.

The full scale range then is $N \times (T\text{max} - T\text{min})$. For n-bit resolution

$$1\ \text{LSB} = \frac{N \times (T\text{max} - T\text{min})}{2^n}.$$

Therefore, the number of periods over which the elapsed time should be measured is

$$N = \frac{2^n}{T\text{max} - T\text{min}}.$$

However, N must also be an integer. It is logical to evaluate the above formula (do not forget that *T*max and *T*min have to be in machine cycles) and select for N the next higher integer. This selection gives a period measurement that has somewhat more than n-bit resolution, which may or may not be acceptable, depending on the overall requirements of the

IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. IE-32, NO. 4, NOVEMBER 1985

system. It can be scaled back to n-bit resolution, if necessary, by the following computation:

$$\text{reported value} = \frac{NT - NT\text{min}}{NT\text{max} - NT\text{min}}$$

where NT is the elapsed time measured over N periods.

The computation can be done in math if a suitable divide routine is available in the software. For 8-bit resolution it is entirely reasonable to find the reported value in a look-up table, which would take up somewhat more than one page in ROM. In fact, the look-up table would contain $NT\text{max} - NT\text{min}$ entries.

For example, suppose we want 8-bit resolution in the measurement of the period of a signal whose frequency varies from 5 to 10 kHz. If the clock frequency is 12 MHz, then $T\text{max}$ is (12 000 kHz)/(12 × 5 kHz) = 200 machine cycles, and $T\text{min}$ is 100 machine cycles. The timer needs to be on then for $N = 2.56$ periods, according to the formula. Using $N = 3$ periods will give maximum and minimum NT values of 600 and 300 machine cycles. This is somewhat more than 8-bit resolution. It can be scaled to 8 bits with a 300-byte look-up table, if desired.

To implement the measurement, one timer is used to measure the elapsed time NT. Enabling its interrupt is optional. The timer interrupt could be used to indicate a short or open in the transducer circuit.

Then the transducer is connected to one of the external interrupt pins (INT0 or INT1), and this interrupt is configured to the transition-activated mode. In the transition-activated mode every 1-to-0 transition in the transducer output will generate an interrupt. The interrupt routine counts transducer pulses, and when it gets to the predetermined N, it reads and clears the timer. For example

```
    DJNZ    PULSES,OUT
    MOV     PULSES,#N_PERIODS
    (Read and clear timer.)
OUT: RETI
```

If other interrupts are also to be enabled, the one connected to the transducer should be set to Priority 1, and the others to Priority 0. This is to control the interrupt response time. The response time will not affect the measurement if it is the same for every measurement. Variations in the response time will, however, affect the measurement. Setting the pulse-counter interrupt to Priority 1 and all others to Priority 0 will minimize variations in the response time. The response time will then be limited to range from 3 to 8 machine cycles.

VII. Pulsewidth Measurements

The 8051 timers have an operating mode which is particularly suited to pulsewidth measurements, and may be useful here if the transducer has a fixed duty cycle, or if the transducer output is pulsewidth modulated instead of frequency modulated by the measurand.

In this mode the timer is turned on by the on-chip circuitry in response to an input high at the external interrupt pin, and off by an input low. The external interrupt itself is enabled, so the same 1-to-0 transition from the transducer that turns off the timer also generates an interrupt. The interrupt routine would then read and reset the timer.

The advantage of this method is that the transducer signal has direct access to the timer gate, with the result that variations in the interrupt response time cease to be a factor. The timer can be read and cleared any time before the next high in the transducer output.

VIII. Deriving Frequency From a Period Measurement

We now consider the problem of measuring the transducer frequency to n-bit resolution by deriving it from a direct measurement of the period. The advantage of this technique is speed. It is always faster to measure period than frequency. But it is important to end up with a frequency value that has the same resolution and accuracy as a directly measured frequency. Two questions need to be addressed.

1) To achieve n-bit resolution in the calculated frequency, how much resolution is required in the period?
2) Having measured the period to the required resolution, what is the most efficient way to calculate the frequency?

These questions will be addressed one at a time.

IX. Resolution Requirements

In general, n-bit resolution in the frequency derivation requires somewhat more than n-bit resolution in the period measurement. How much more? It will be demonstrated presently that if the transducer frequency varies over a 2-to-1 range, the frequency can be calculated with n-bit resolution from period measurements that have $(n + 1)$-bit resolution.

The more practical form of the question is over how many periods (N) must the transducer signal be sampled to obtain the required resolution in F? And so, we commence a calculation of N.

The basic calculation of frequency from $N \times T$ (which we shall call NT) is straightforward:

$$F = N/(NT).$$

The relationship between an increment dF in the calculated frequency due to an increment $d(NT)$ in the measured period is, therefore,

$$dF = -\frac{N}{(NT)^2}\, d(NT)$$

$$= -\frac{F^2}{N}\, d(NT).$$

This equation says the value of an LSB in the calculated frequency is $(F^2)/N \times$ the value of an LSB in NT. Then the maximum value that an LSB in the calculated frequency can have is $(F\text{max})^2/N \times$ the value of an LSB in NT. For the calculated frequency to have n-bit resolution over the entire range of frequencies, the value of its LSB must never exceed $(F\text{max} - F\text{min})/2^n$. Therefore, the measurement requires

$$\frac{(F\text{max})^2}{N} \times (1 \text{ LSB in } NT) \leq \frac{F\text{max} - F\text{min}}{2^n}.$$

270434-3

The required resolution in NT is, therefore,

$$1 \text{ LSB in } NT \le \frac{N \times (F\max - F\min)}{2^n \times (F\max)^2}.$$

Now, to say that NT is measured with m-bit resolution means

$$1 \text{ LSB in } NT = \frac{N \times (1/F\min - 1/F\max)}{2^m}.$$

Substituting this value for 1 LSB into the required resolution and solving for 2^m yields

$$2^m \ge \frac{F\max}{F\min} \times 2^n.$$

Then the requirement on m is

$$m \ge n + \frac{\ln (F\max/F\min)}{\ln (2)}.$$

It can be stated with some certainty, then, that if the transducer frequency varies over a range of 2-to-1, the frequency can be calculated with 8-bit resolution from a period measurement that has 9-bit resolution. If the frequency variation is less than 2-to-1, another full bit of resolution in the period measurement is not needed.

To obtain m-bit resolution in NT, N must satisfy

$$N \ge \frac{2^m}{T\max - T\min}.$$

Substituting for 2^m, and using $T\max = 1/F\min$ and $T\min = 1/F\max$, gives the result that

$$N \ge \frac{(F\max)^2}{F\max - F\min} \times 2^n.$$

It should be noted that the units of frequency here are periods/machine cycle, since the 8051 measures time by counting machine cycles. The conversion factor between Hz and periods/machine cycle is 12/(clock frequency). So the requirement on N can also be written

$$N \ge \frac{F\max}{F\max - F\min} \times \frac{F\max}{F\text{xtal}} \times 12 \times 2^n$$

where Fxtal is the 8051 clock frequency in the same units as Fmax and Fmin. This is the number of transducer pulses over which the transducer signal must be sampled to achieve the required solution in F.

For example, suppose that 8-bit resolution is required in F, where Fmax = 10 kHz and Fmin = 5 kHz, and that Fxtal = 12 MHz. Then the above calculation shows that $N = 6$ periods gives sufficient resolution in the period measurement to satisfy the resolution requirement in F. Six periods will take 0.6–1.2 ms of sampling time, on that frequency range. Recall that the sample time for a direct frequency measurement of the same signal and to the same resolution was earlier calculated to be 51.2 ms.

X. Computing the Frequency from the Period

The period measurement leaves one with a 16-bit integer, which is $N \times T$ (or NT) in machine cycles. The conversion to frequency is straightforward:

$$F = N/(NT) \text{ periods/machine cycle.}$$

The quantity of interest is probably not F, but a normalized measure of the amount by which F exceeds its minimum acceptable value. This quantity represents, through the transducer's transfer function, the "reported value" of the measurand, and this quantity is an n-bit integer whose value ranges from 0 (all bits = 0) to full scale (all bits = 1). This normalized frequency is

$$f = \frac{F - F\min}{F\max - F\min} = \frac{F\min}{F\max - F\min} \times (F/F\min - 1)$$

Using $F = N/(NT)$ and $F\min = N/(NT\max)$ allows the normalized frequency to be written

$$f = \frac{F\min}{F\max - F\min} \times \frac{NT\max - NT}{NT}.$$

To get a handle on what kinds of numbers are involved here, consider the situation where 8-bit resolution is required in f, and in which Fxtal = 12 MHz, Fmax = 10 kHz, and Fmin = 5 kHz. We have previously determined that for this set of conditions, $N = 6$ periods gives sufficient resolution in the period measurement to satisfy the resolution requirement in F (and in f). With a 12 MHz clock frequency, Tmax in machine cycles is (12 000 kHz)/(12 × 5 kHz) = 200 machine cycles, so NTmax is 6 × 200 = 1200 machine cycles. The calculation for f then becomes

$$f = \frac{1200 - NT}{NT}.$$

The minimum acceptable value that NT can have is ($N \times T\min + 1$), where Tmin = (12 000 kHz)/(12 × 10 kHz) = 100 machine cycles. Then $N \times T\min = 6 \times 100 = 600$ machine cycles. The allowable values for NT are then 601–1200 machine cycles, a total of 599 different values.

The fastest way to "calculate" f would be with a 599-byte look-up table. This method has the added advantage that nonlinearities in the transfer function between frequency and measurand can be built into the table. Look-up tables are facilitated in the 8051 by the MOVC A,@A + PC, and MOVC A,@A + DPTR instructions. DPTR is a 16-bit "data pointer" register in the 8051. Its two bytes can be individually addressed as DPL (low byte) and DPH (high byte).

In the example under discussion, it will be necessary to load DPTR with the address of the first byte of the look-up table, less 601, plus the 2-byte value of NT. The software that accesses the table might then take the following form:

```
TABLE EQU (address of first table entry)
```

270434–4

```
                    DIVIDE ROUTINE

This divide routine calculates

                          numerator
                quotient  ---------
                          denominator

in which numerator and denominator are integers and
numerator > denominator. Quotient is of the form

        quotient = 0. q1 q2 q3 ... qN

where qn is the coefficient of 2**(-n). The procedure is

    numerator = 2 * numerator
    n = 1
    while n <= N+1 do
        if numerator >= denominator then qn = 0 else qn = 1
        if qn = 0 then numerator = 2 * numerator
                  else numerator = 2 * numerator - denominator
        increment n
    end_while
```

Fig. 1. A divide algorithm.

```
MOV   A,#LOW(TABLE - 601)
ADD   A,NT_LO
MOV   DPL,A
MOV   A,#HIGH(TABLE - 601)
ADDC  A,NT_HI
MOV   DPH,A
CLR   A
MOVC  A,@A + DPTR.
```

At this point the accumulator contains the 8-bit value of f.

It is perfectly reasonable to decide that a 599-byte look-up table is unwieldy. Its advantages are speed and built-in error correction. But a reasonably fast divide algorithm can be written to this specific purpose, making use of *a priori* knowledge about the sizes of the numbers that are involved in the computation. It helps to know that in this example the numerator is never going to be larger than 599 and the denominator is always greater than the numerator.

A complete discussion of divide routines is beyond the scope of this paper, but a suitable divide algorithm for this specific application is shown in Fig. 1. Reference [1] calls this the Restoring division algorithm. It is particularly well suited to the 8051, because "<" comparisons are greatly facilitated by the 8051's CJNE (compare and jump if not equal) instruction. CJNE A,B,rel, executes a relative jump if A does not equal B. More importantly to this application, the instruction sets the carry flag if $A < B$.

XI. Accuracy and Resolution

The accuracy with which the 8051 will measure the frequency or period of the transducer signal depends on two things: the accuracy of the clock oscillator and variations in the interrupt response time.

Since the clock signal is normally generated by a crystal oscillator, the oscillator accuracy normally far exceeds the quantizing error inherent in the finite (n-bit) resolution.

As was previously mentioned, interrupt response time does not introduce an error into the measurement itself, but variations in the interrupt response time can. Interrupt response time in the 8051 can vary from 3 to 8 machine cycles, depending on what instruction is in progress at the time the interrupt is generated. This would represent an error of ± 5 counts in the measured value of NT during a period measurement. An error of ± 5 counts in NT does not necessarily translate to ± 5 LSB's in the final result, but it might still represent an error that exceeds the resolution.

In a direct frequency measurement variations in the interrupt response time would represent an error of ± 5 μs in the sample time.

If these kinds of errors are unacceptable there are ways to deal with them. In period measurements, if the duty cycle of the transducer is constant, the pulsewidth measurement technique, previously described, can be used. Its advantage is that it gates the timer off when the interrupt is generated, rather than when the interrupt is responded to.

In other cases one can simply increase the sample time above the minimum required to obtain the desired resolution. For example, if the measurement requires 8-bit resolution, one can design the software for an 11-bit resolution and truncate the result to 8 bits.

References

[1] Davio *et al.*, *Digital Systems with Algorithm Implementation*. New York: Wiley, 1983.

270434-5

8051
SOFTWARE PACKAGES

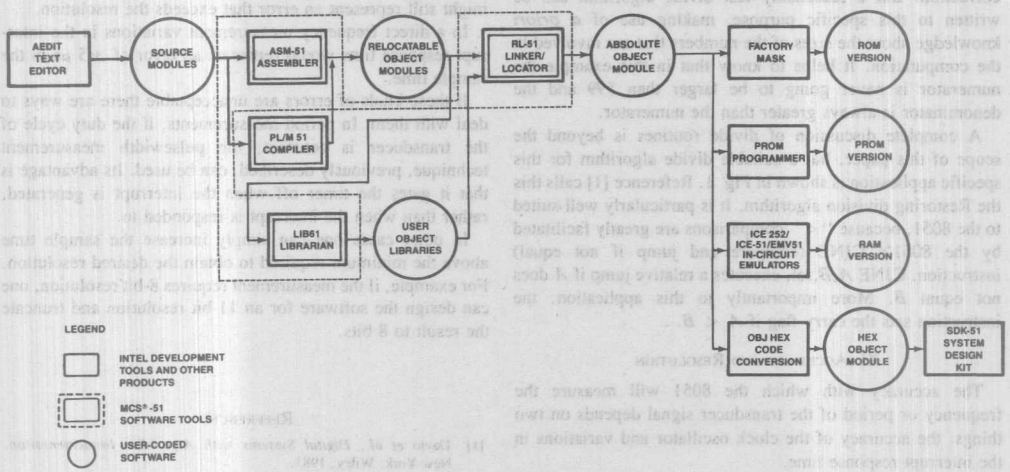
- **Choice of hosts: PCDOS 3.0 based IBM* PC XT/AT*, iRMX®86, iPDS™ System, Series II, Series III, and Series IV**
- **Supports all members of the Intel MCS® -51 architecture**

PL/M51 Software Package Contains the following:

- **PL/M51 Compiler which is designed to support all phases of software implementation**
- **RL51 Linker and Relocator which enables programmers to develop software in a modular fashion**
- **LIB51 Librarian which lets programmers create and maintain libraries of software object modules**

8051 Software Development Package Contains the following:

- **8051 Macro Assembler which gives symbolic access to 8051 hardware features**
- **RL51 Linker and Relocator program which links modules generated by the assembler**
- **LIB51 Librarian which lets programmers create and maintain libraries of software object modules**



162771-1

Figure 1. MCS® -51 Program Development Process

*IBM and AT are registered trademarks of International Business Machine Corporation.

October 1987
Order Number: 162771-005

PL/M 51 SOFTWARE PACKAGE

- **High-level programming language for the Intel MCS® -51 single-chip microcomputer family**
- **Compatible with PL/M 80 assuring MCS® -80/85 design portability**
- **Enhanced to support boolean processing**
- **Tailored to provide an optimum balance among on-chip RAM usage, code size and code execution time**
- **Produces relocatable object code which is linkable to object modules generated by all other 8051 translators**
- **Allows programmer to have complete control of microcomputer resources**
- **Extends high-level language programming advantages to microcontroller software development**
- **Improved reliability, lower maintenance costs, increased programmer productivity and software portability**
- **Includes the linking and relocating utility and the library manager**
- **Supports all members of the Intel MCS® -51 architecture**

PL/M 51 is a structured, high-level programming language for the Intel MCS-51 family of microcomputers. The PL/M 51 language and compiler have been designed to support the unique software development requirements of the single-chip microcomputer environment. The PL/M language has been enhanced to support Boolean processing and efficient access to the microcomputer functions. New compiler controls allow the programmer complete control over what microcomputer resources are used by PL/M programs.

PL/M 51 is largely compatible with PL/M 80 and PL/M 86. A significant proportion of existing PL/M software can be ported to the MCS-51 with modifications to support the MCS-51 architecture. Existing PL/M programmers can start programming for the MCS-51 with a small relearning effort.

PL/M 51 is the high-level alternative to assembly language programming for the MCS-51. When code size and code execution speed are not critical factors, PL/M 51 is the cost-effective approach to developing reliable, maintainable software.

The PL/M 51 compiler has been designed to support efficiently all phases of software implementation with features like a syntax checker, multiple levels of optimization, cross-reference generation and debug record generation.

ICE™ 5100, ICE 51, and EMV51 are available for on-target debugging.

Software available for PC DOS 3.0 based IBM* PC XT/AT* Systems.

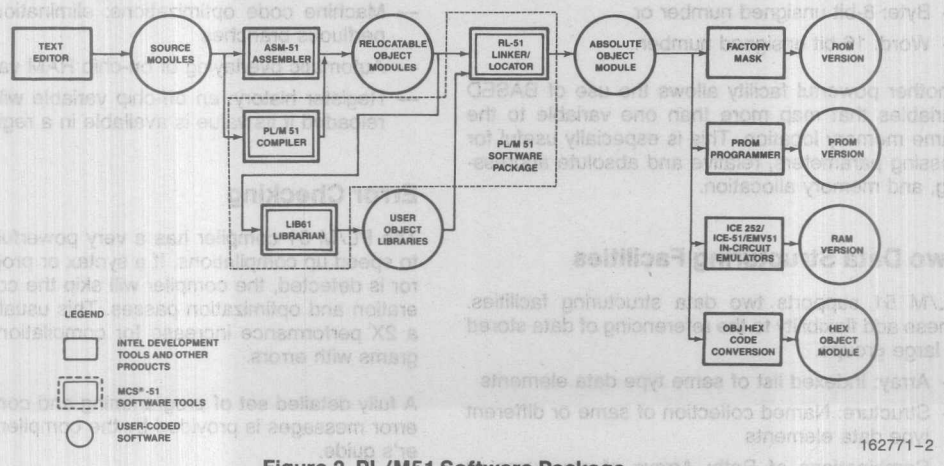


Figure 2. PL/M51 Software Package

PL/M 51 COMPILER

FEATURES

Major features of the Intel PL/M 51 compiler and programming language include:

Structured Programming

PL/M source code is developed in a series of modules, procedures, and blocks. Encouraging program modularity in this manner makes programs more readable, and easier to maintain and debug. The language becomes more flexible, by clearly defining the scope of user variables (local to a private procedure, for example).

Language Compatibility

PL/M 51 object modules are compatible with object modules generated by all other MCS-51 translators. This means that PL/M programs may be linked to programs written in any other MCS-51 language.

Object modules are compatible with In-Circuit Emulators and Emulation Vehicles for MCS-51 processors: the DEBUG compiler control provides these tools with symbolic debugging capabilities.

Supports Three Data Types

PL/M makes use of three data types for various applications. These data types range from one to sixteen bits and facilitate various arithmetic, logic, and address functions:

— Bit: a binary digit

— Byte: 8-bit unsigned number or,

— Word: 16-bit unsigned number.

Another powerful facility allows the use of BASED variables that map more than one variable to the same memory location. This is especially useful for passing parameters, relative and absolute addressing, and memory allocation.

Two Data Structuring Facilities

PL/M 51 supports two data structuring facilities. These add flexibility to the referencing of data stored in large groups.

— Array: Indexed list of same type data elements

— Structure: Named collection of same or different type data elements

— Combinations of Both: Arrays of structures or structures of arrays.

Interrupt Handling

A procedure may be defined with the INTERRUPT attribute. The compiler will generate code to save and restore the processor status, for execution of the user-defined interrupt handler routines.

Compiler Controls

The PL/M 51 compiler offers controls that facilitate such features as:

— Including additional PL/M 51 source files from disk

— Cross-reference

— Corresponding assembly language code in the listing file

Program Addressing Control

The PL/M 51 compiler takes full advantage of program addressing with the ROM (SMALL/MEDIUM/LARGE) control. Programs with less than 2 KB code space can use the SMALL or MEDIUM option to generate optimum addressing instructions. Larger programs can address over the full 64 KB range.

Code Optimization

The PL/M 51 compiler offers four levels of optimization for significantly reducing overall program size.

— Combination or "folding" of constant expressions; "Strength reductions" (a shift left rather than multiply by 2)

— Machine code optimizations; elimination of superfluous branches

— Automatic overlaying of on-chip RAM variables

— Register history: an off-chip variable will not be reloaded if its value is available in a register.

Error Checking

The PL/M 51 compiler has a very powerful feature to speed up compilations. If a syntax or program error is detected, the compiler will skip the code generation and optimization passes. This usually yields a 2X performance increase for compilation of programs with errors.

A fully detailed set of programming and compilation error messages is provided by the compiler and user's guide.

BENEFITS

PL/M 51 is designed to be an efficient, cost-effective solution to the special requirements of MCS-51 Microsystem Software Development, as illustrated by the following benefits of PL/M use:

Low Learning Effort

PL/M 51 is easy to learn and to use, even for the novice programmer.

Earlier Project Completion

Critical projects are completed much earlier than otherwise possible because PL/M 51, a structured high-level language, increases programmer productivity.

Lower Development Cost

Increases in programmer productivity translate immediately into lower software development costs because less programming resources are required for a given programmed function.

Increased Reliability

PL/M 51 is designed to aid in the development of reliable software (PL/M programs are simple statements of the program algorithm). This substantially reduces the risk of costly correction of errors in systems that have already reached full production status, as the more simply stated the program is, the more likely it is to perform its intended function.

Easier Enhancements and Maintenance

Programs written in PL/M tend to be self-documenting, thus easier to read and understand. This means it is easier to enhance and maintain PL/M programs as the system capabilities expand and future products are developed.

RL51 LINKER AND RELOCATOR

- **Links modules generated by the assembler and the PL/M compiler**
- **Locates the linked object to absolute memory locations**
- **Enables modular programming of software-efficient program development**
- **Modular programs are easy to understand, maintainable and reliable**

The MCS-51 linker and relocator (RL51) is a utility which enables MCS-51 programmers to develop software in a modular fashion. The utility resolves all references between modules and assigns absolute memory locations to all the relocatable segments, combining relocatable partial segments with the same name.

With this utility, software can be developed more quickly because small functional modules are easier to understand, design and test than large programs.

The total number of allowed symbols in user-developed software is very large because the assembler number of symbols' limit applies only per module, not to the entire program. Therefore programs can be more readable and better documented. RL51 can be invoked either manually or through a batch file for improved productivity.

Modules can be saved and used on different programs. Therefore the software investment of the customer is maintained.

RL51 produces two files. The absolute object module file can be directly executed by the MCS-51 family. The listing file shows the results of the link/locate process.

LIB51 LIBRARIAN

The LIB51 utility enables MCS-51 programmers to create and maintain libraries of software object modules. With this utility, the customer can develop standard software modules and place them in libraries, which programs can access through a standard interface. When using object libraries, the linker will call only object modules that are required to satisfy external references.

Consequently, the librarian enables the customer to port and reuse software on different projects—thereby maintaining the customer's software investment.

ORDERING INFORMATION

Order Code	Operating Environment
D86PLM51	PL/M51 Software for PC DOS 3.0 Systems
R86PLM51	PL/M51 Software for iRMX 86 Systems

Documentation Package

PL/M 51 User's Guide

MCS-51 Utilities User's Guide

SUPPORT:

Hotline Telephone Support, Software Performance Report (SPR), Software Updates, Technical Reports, and monthly Technical Newsletters are available.

8051 SOFTWARE DEVELOPMENT PACKAGE

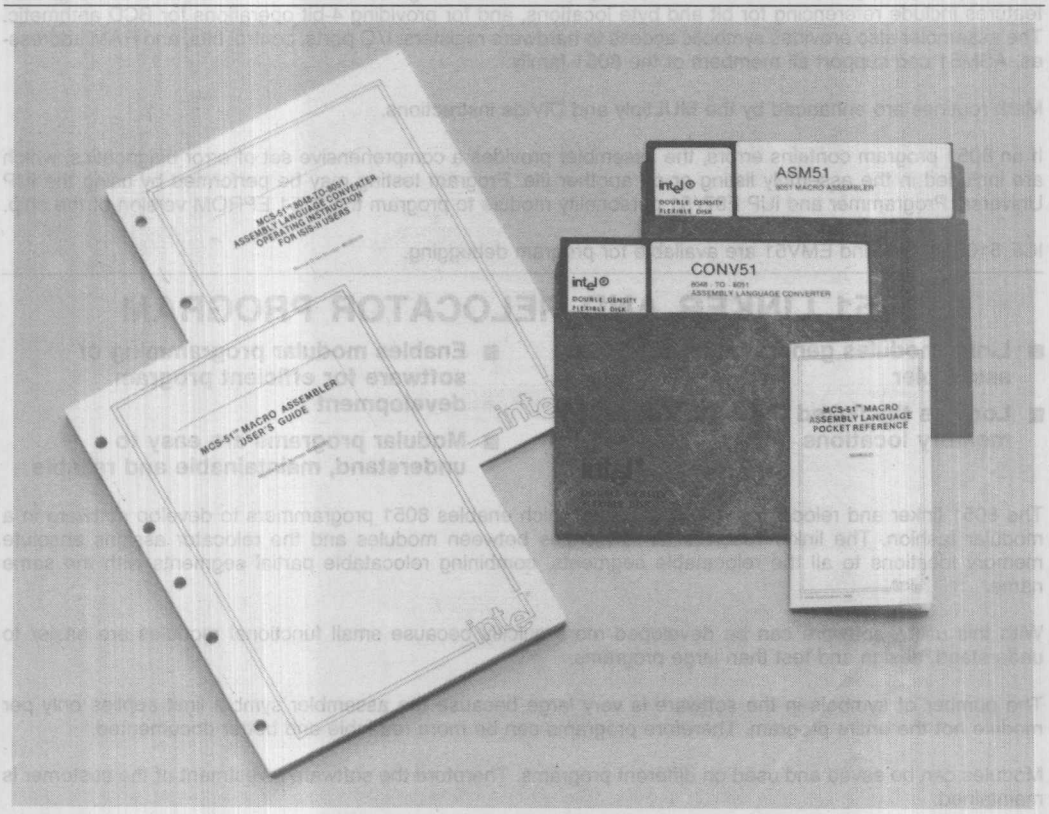
- **Symbolic relocatable assembly language programming for 8051 microcontrollers**
- **Extends Intellec® Microcomputer Development System to support 8051 program development**
- **Produces Relocatable Object Code which is linkable to other 8051 Object Modules**
- **Encourage modular program design for maintainability and reliability**
- **Macro Assembler features conditional assembly and macro capabilities**
- **Supports all members of the Intel MCS® 51 architecture**

The 8051 software development package provides development system support for the powerful 8051 family of single chip microcomputers. The package contains a symbolic macro assembler and relocation/linkage utilities.

The assembler produces relocatable object modules from 8051 macro assembly language instructions. The object code modules can be linked and located to absolute memory locations. This absolute object code may be used to program the 8751 EPROM version of the chip. The assembler output may also be debugged using the new family of ICE 5100 emulators or with the ICE-51™ in-circuit emulator.

The converter translates 8048 assembly language instructions into 8051 source instructions to provide software compatibility between the two families of microcontrollers.

Software available for PC DOS 3.0 based IBM* PC XT/AT Systems.



162771-3

8051 MACRO ASSEMBLER

- **Supports 8051 family program development on Intellec® Microcomputer Development Systems**
- **Gives symbolic access to powerful 8051 hardware features**
- **Produces object file, listing file and error diagnostics**
- **Object files are linkable and locatable**
- **Provides software support for many addressing and data allocation capabilities**
- **Symbolic Assembler supports symbol table, cross-reference, macro capabilities, and conditional assembly**

The 8051 Macro Assembler (ASM51) translates symbolic 8051 macro assembly language modules into linkable and locatable object code modules. Assembly language mnemonics are easier to program and are more readable than binary or hexadecimal machine instructions. By allowing the programmer to give symbolic names to memory locations rather than absolute addresses, software design and debug are performed more quickly and reliably. Furthermore, since modules are linkable and relocatable, the programmer can do his software in modular fashion. This makes programs easy to understand, maintainable and reliable.

The assembler supports macro definitions and calls. This is a convenient way to program a frequently used code sequence only once. The assembler also provides conditional assembly capabilities.

Cross referencing is provided in the symbol table listing, showing the user the lines in which each symbol was defined and referenced.

ASM51 provides symbolic access to the many useful addressing features of the 8051 architecture. These features include referencing for bit and byte locations, and for providing 4-bit operations for BCD arithmetic. The assembler also provides symbolic access to hardware registers, I/O ports, control bits, and RAM addresses. ASM51 can support all members of the 8051 family.

Math routines are enhanced by the MULtiply and DIVide instructions.

If an 8051 program contains errors, the assembler provides a comprehensive set of error diagnostics, which are included in the assembly listing or on another file. Program testing may be performed by using the iUP Universal Programmer and iUP F87/51 personality module to program the 8751 EPROM version of the chip.

ICE 5100, ICE51 and EMV51 are available for program debugging.

RL51 LINKER AND RELOCATOR PROGRAM

- **Links modules generated by the assembler**
- **Locates the linked object to absolute memory locations**
- **Enables modular programming of software for efficient program development**
- **Modular programs are easy to understand, maintainable and reliable**

The 8051 linker and relocator (RL51) is a utility which enables 8051 programmers to develop software in a modular fashion. The linker resolves all references between modules and the relocator assigns absolute memory locations to all the relocatable segments, combining relocatable partial segments with the same name.

With this utility, software can be developed more quickly because small functional modules are easier to understand, design and test than large programs.

The number of symbols in the software is very large because the assembler symbol limit applies only per module not the entire program. Therefore programs can be more readable and better documented.

Modules can be saved and used on different programs. Therefore the software investment of the customer is maintained.

RL51 produces two files. The absolute object module file can be directly executed by the 8051 family. The listing file shows the results of the link/locate process.

LIB51 LIBRARIAN

The LIB51 utility enables MCS-51 programmers to create and maintain libraries of software object modules. With this utility, the customer can develop standard software modules and place them in libraries, which programs can access through a standard interface. When using object libraries, the linker will call only object modules that are required to satisfy external references.

Consequently, the librarian enables the customer to port and reuse software on different projects—thereby maintaining the customer's software investment.

ORDERING INFORMATION

Order Code	Operating Environment
D86ASM51	8051 Assembler for PCDOS 3.0 Systems
R86ASM51	8051 Assembler for iRMX 86 Systems

Documentation Package:

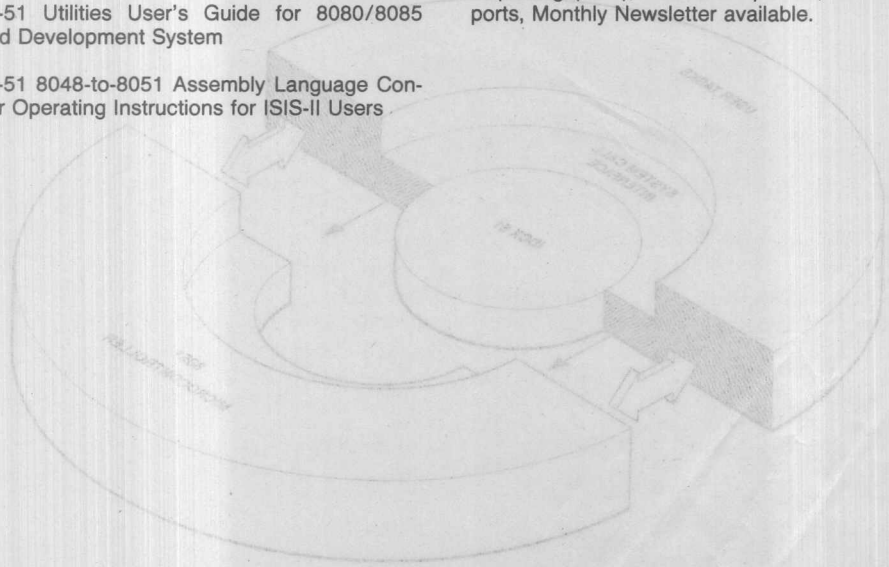
MCS-51 Macro Assembler User's Guide

MCS-51 Utilities User's Guide for 8080/8085 Based Development System

MCS-51 8048-to-8051 Assembly Language Converter Operating Instructions for ISIS-II Users

SUPPORT:

Hotline Telephone Support, Software Performance Reporting (SPR), Software Updates, Technical Reports, Monthly Newsletter available.



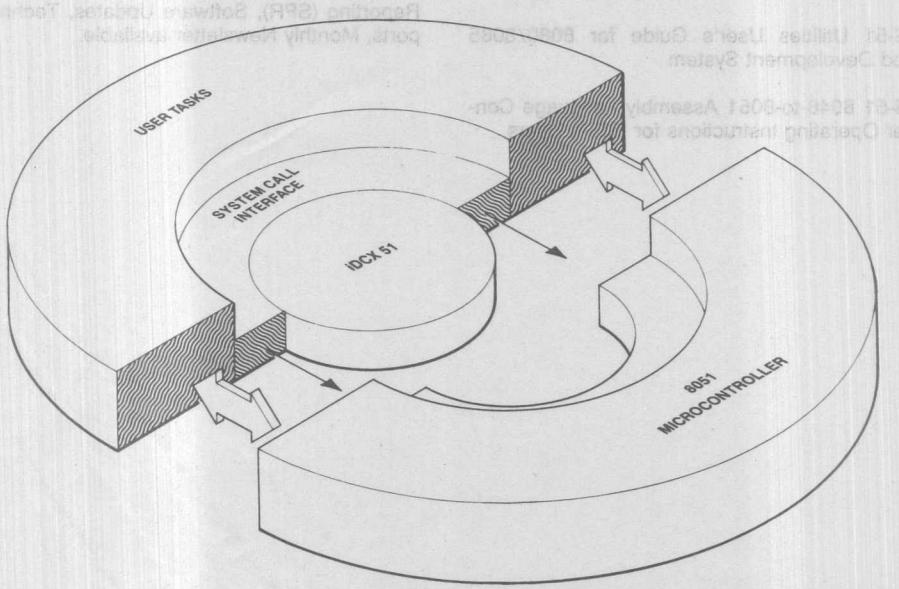
iDCX 51
DISTRIBUTED CONTROL EXECUTIVE

- **Supports MCS®-51 and RUPI™-44 Familes of 8-Bit Microcontrollers**
- **Real-Time, Multitasking Executive — Supports up to 8 Tasks at Four Priority Levels**
- **Local and Remote Task Communication**
- **Small—2.2K Bytes**
- **Reliable**
- **Simple User Interface**
- **Dynamic Reconfiguration Capability**
- **Compatible with BITBUS™/Distributed Control Modules (iDCM) Product Line**

The iDCX 51 Executive is compact, easy to use software for development and implementation of applications using the high performance 8-bit family of 8051 microcontrollers, including the 8051, 8044, and 8052. Like the 8051 family, the iDCX 51 Executive is tuned for real-time control applications requiring manipulation and scheduling of more than one task, and fast response to external stimuli.

The MCS-51 microcontroller family coupled with iDCX 51 is a natural combination for applications such as data acquisition and monitoring, process control, robotics, and machine control. The iDCX 51 Executive can significantly reduce applications development time, particularly BITBUS distributed control environments.

The iDCX 51 Executive is available in two forms, either as configurable software on diskette or as preconfigured firmware within the 8044 BEM BITBUS microcontroller.



280176-1

Figure 1. iDCX 51 Distributed Control Executive

*XENIX™ is a trademark of Microsoft Corporation.

October 1987
Order Number: 280176-003

MICROCONTROLLER SUPPORT

The iDCX 51 Executive is designed to support the MCS-51 and RUPI-44 families of 8-bit microcontrollers. MCS-51 microcontrollers that are supported include the 8051, 80C51, 8052, 8031, 8032, and 8751 devices. The RUPI-44 microcontrollers include the 8044, 8344, and 8744 devices. All of these microcontrollers share a common 8051 core.

ARCHITECTURE

Real-time and Multitasking

Real-time control applications must be responsive to the external environment and typically involve the execution of more than one activity (task or set of tasks) in response to different external stimuli. Control of an industrial drying process is an example. This process could require monitoring of multiple temperatures and humidity; control of fans, heaters, and motors that must respond accordingly to a variety of inputs. The iDCX 51 Executive fully supports applications requiring response to stimuli as they occur, i.e., in real-time. This real-time response is supported for multiple tasks often needed to implement a control application.

Some of the facilities precisely tailored for development and implementation of real-time control application systems provided by the iDCX 51 Executive are: task management, interrupt handling, message passing, and when integrated with communications support, message passing with different microcontrollers. Also, the iDCX 51 Executive is driven by events: interrupts, timers, and messages ensuring the application system always responds to the environment appropriately.

Task Management

A task is a program defined by the user to execute a particular control function or functions. Multiple programs or tasks may be required to implement a particular function such as "controlling Heater 1". The iDCX 51 Executive recognizes three different task states as one of the mechanisms to accomplish scheduling of up to eight tasks. Figure 2 illustrates the different task states and their relationship to one another.

The scheduling of tasks is priority based. The user can prioritize tasks to reflect their relative importance within the overall control scheme. For instance, if Heater 1 must go off line prior to Heater 2 then the task associated with Heater 1 shutdown could be assigned a higher priority ensuring the correct shutdown sequence. The RQ WAIT system call is also a scheduling tool. In this example the task implementing Heater 2 shutdown could include an instruction to wait for completion of the task that implements Heater 1 shutdown.

The iDCX 51 Executive allows for PREEMPTION of a task that is currently being executed. This means that if some external event occurs such as a catastrophic failure of Heater 1, a higher priority task associated with the interrupt, message, or timeout resulting from the failure will preempt the running task. Preemption ensures the emergency will be responded to immediately. This is crucial for real-time control application systems.

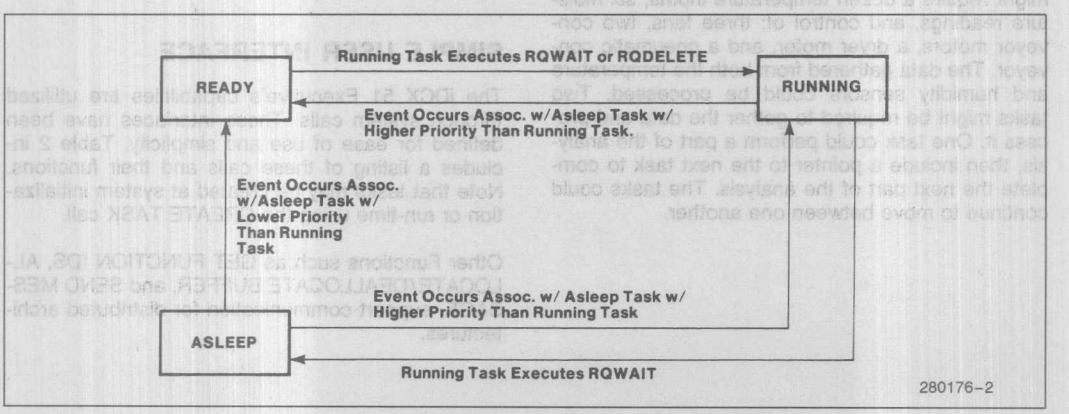


Figure 2. Task State Transition Diagram

Interrupt Handling

The iDCX 51 Executive supports five interrupt sources as shown in Table 1. Four of these interrupt sources, excluding timer 0, can be assigned to a task. When one of the interrupts occurs the task associated with it becomes a running task (if it were the highest priority task in a ready state). In this way, the iDCX 51 Executive responds to a number of internal and external stimuli including time intervals designed by the user.

Table 1. iDCX 51 Interrupt Sources

Interrupt Source	Interrupt Number
External Request 0	00H
Timer 0	01H
External Request 1	02H
Timer 1	03H
Internal Serial Port 1	04H

Message Passing

The iDCX 51 Executive allows tasks to interface with one another via a simple message passing facility. This message passing facility can be extended to different processors when communications support is integrated within a BITBUS/iDCM system, for example. This facility provides the user with the ability to link different functions or tasks. Linkage between tasks/functions is typically required to support development of complex control applications with multiple sensors (input variables) and drivers (output variables). For instance, the industrial drying process might require a dozen temperature inputs, six moisture readings, and control of: three fans, two conveyor motors, a dryer motor, and a pneumatic conveyor. The data gathered from both the temperature and humidity sensors could be processed. Two tasks might be required to gather the data and process it. One task could perform a part of the analysis, then include a pointer to the next task to complete the next part of the analysis. The tasks could continue to move between one another.

REMOTE TASK COMMUNICATION

The iDCX 51 Executive system calls can support communication to tasks on remote controllers. This feature makes the iDCX 51 Executive ideal for applications using distributed architectures. Providing communication support saves significant application development time and allows for more effective use of this time. Intel's iDCM product line combines hardware and software to provide this function.

In an iDCM system, communication between nodes occurs via the BITBUS microcontroller interconnect. The BITBUS microcontroller interconnect is a high performance serial control bus specifically intended for use in applications built on distributed architectures. The iDCX 51 Executive provides BITBUS support.

BITBUS™/iDCM COMPATIBLE

A pre-configured version of the iDCX 51 Executive implements the BITBUS message format and provides all iDCX 51 facilities mentioned previously: task management, interrupt handling, and message passing. This version of the Executive is supplied in firmware on the 8044 BEM with the iDCM hardware products: the iSBX™ 344A BITBUS Controller MULTIMODULE™; the iDCX 344A BITBUS controller board for the PC; and the iRCB boards.

Designers who want to use the iDCX executive on an Intel BITBUS board should purchase either DCS110 or DSC120 BITBUS software. Both of these products include an interface library to iDCX 51 procedures and other development files. It is not necessary to purchase the iDCX 51 Executive.

SIMPLE USER INTERFACE

The iDCX 51 Executive's capabilities are utlilized through system calls. These interfaces have been defined for ease of use and simplicity. Table 2 includes a listing of these calls and their functions. Note that tasks may be created at system initialization or run-time using the CREATE TASK call.

Other Functions such as GET FUNCTION IDS, ALLOCATE/DEALLOCATE BUFFER, and SEND MESSAGE, support communication for distributed architectures.

Table 2. iDCX 51 System Calls

Call Name	Description
TASK MANAGEMENT CALLS	
RQ$CREATE$TASK	Create and schedule a new task.
RQ$DELETE$TASK	Delete specified task from system.
RQGETFUNCTION$IDS	Obtain the function IDs of tasks currently in the system.
RQ$ALLOCATE	Obtain a message buffer from the system buffer pool.
RQ$DEALLOCATE	Return a message buffer to the system buffer pool.
RQ$SEND$MESSAGE	Send a message to specified task.
RQ$WAIT	Wait for a message event.
MEMORY MANAGEMENT CALLS	
RQGETMEM	Get available system memory pool memory.
RQ$RELEASE$MEM	Release system memory pool memory.
INTERRUPT MANAGEMENT CALLS	
RQ$DISABLE$INTERRUPT	Temporarily disable an interrupt.
RQ$ENABLE$INTERRUPT	Re-enable an interrupt.
RQ$WAIT	Wait for an interrupt event.
TIMER MANAGEMENT CALLS	
RQSETINTERVAL	Establish a time interval.
RQ$WAIT	Wait for an interval event.

Another feature that eases application development is automatic register bank allocation. The Executive will assign tasks to register banks automatically unless a specific request is made. The iDCX 51 Executive keeps track of the register assignments allowing the user to concentrate on other activities.

SYSTEM CONFIGURATION

The user configures an iDCX 51 system simply by specifying the initial set of task descriptors and configuration values, and linking the system via the RL 51 Linker and Locator Program with user programs.

Each task that will be running under control of the executive has an Initial Task Description (ITD) that describes it. The ITD specifies to the executive the amount of stack space to reserve, the priority level of the task (1–4), the internal memory register bank to be associated with the task, the internal or external interrupt associated with the task, and a function ID (assigned by the user) that uniquely labels the task. The ITD can also include a pointer to the ITD for the next task. In this way an ITD "chain" can be formed. For example, if four ITD's are chained together, then when the system is initialized, all four tasks will be put into a READY state. Then, the highest priority task will run.

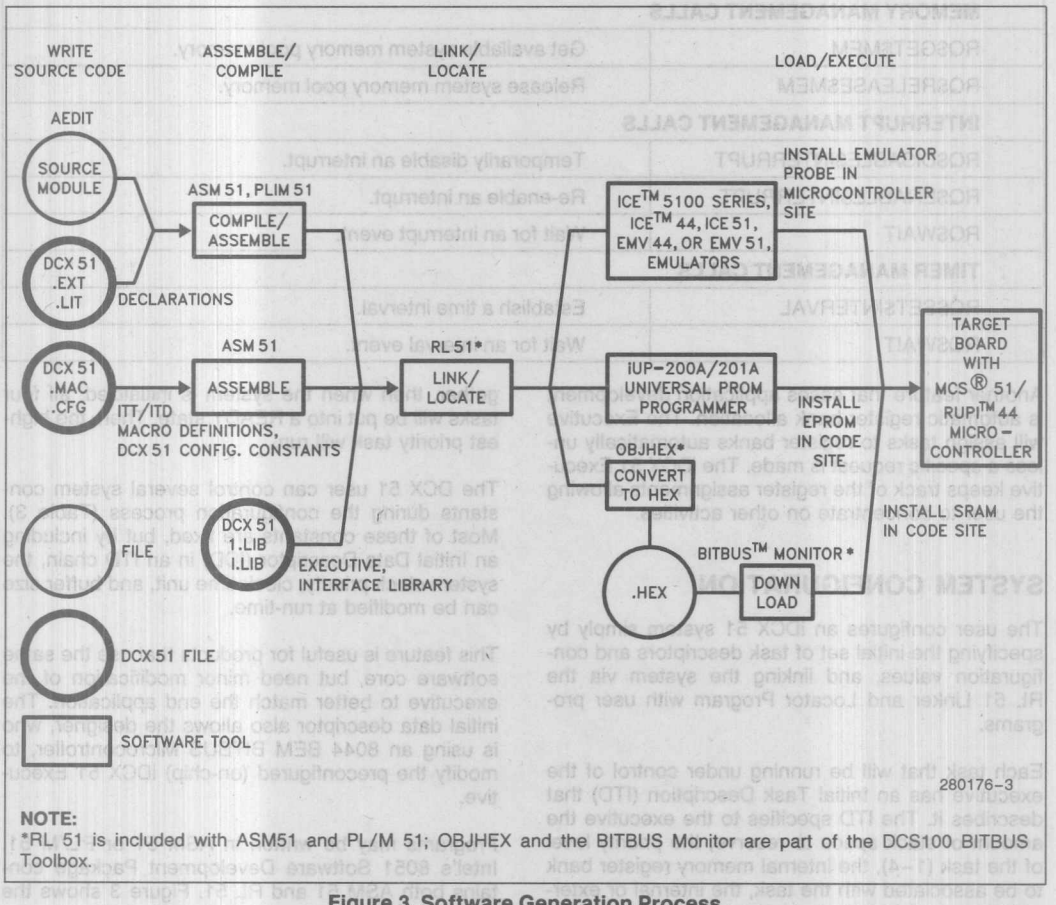
The DCX 51 user can control several system constants during the configuration process (Table 3). Most of these constants are fixed, but by including an Initial Data Descriptor (IDD) in an ITD chain, the system clock priority, clock time unit, and buffer size can be modified at run-time.

This feature is useful for products that use the same software core, but need minor modification of the executive to better match the end application. The initial data descriptor also allows the designer, who is using an 8044 BEM BITBUS Microcontroller, to modify the preconfigured (on-chip) iDCX 51 Executive.

Programs may be written in ASM 51 or PL/M 51. Intel's 8051 Software Development Package contains both ASM 51 and RL 51. Figure 3 shows the software generation process.

Table 3. DCX 51 Configuration Constants

Constant Name	Description
RQ CLOCK PRIORITY	The priority level of the system clock.
RQ CLOCK TICK	The number of time cycles in the system clock basic time unit (a "tick").
RQ FIRST ITD	The absolute address of the first ITD in the ITD chain.
RQ MEM POOL ADR	The start address of the System Memory Pool (SMP) in Internal Data RAM.
RQ MEM POOL LEN	The length of the SMP.
RQ RAM IDD	The absolute RAM address of where iDCX 51 checks for an Initial Data Descriptor (IDD) during initialization.
RQ SYS BUF SIZE	The size, in bytes, of each buffer in the system buffer pool.



NOTE:
*RL 51 is included with ASM51 and PL/M 51; OBJHEX and the BITBUS Monitor are part of the DCS100 BITBUS Toolbox.

Figure 3. Software Generation Process

SOPHISTICATED INTERNAL MEMORY MANAGEMENT

The amount of internal memory available ranges from 128 to 256 bytes depending on the type of microcontroller used.

Internal memory is used for the executive, stack spare for "running" tasks, space for message buffers, and reserved memory for variables storage. Other memory is used for register space. Except for register space, the allocation of internal memory is controlled by the executive, user-specified task/data descriptors and system configuration constants.

To optimize use of this limited resource, iDCX 51 provides dynamic (run-time) memory management.

INITIALIZATION AND DYNAMIC MEMORY MANAGEMENT

At initialization (see Figure 4), the iDCX 51 Executive creates the System Memory Pool (SMP) out of the remaining initial free space (i.e. memory not used by the iDCX 51 Executive or for register space). Next, stack space is created for each of the initial tasks that will be running on the system. If reserved memory is requested (using an IDD), that memory is also set aside. Finally, multiple buffers (size specified during iDCX 51 configuration or using an IDD) are allocated from any remaining memory. These buffers form the System Buffer Pool (SBP) that can be used to create additional stack space or to locate messages sent between tasks.

During run-time, the iDCX 51 Executive dynamically manages this space. If a task is deleted, its stack space is returned to the System Buffer Pool for use by other tasks or as a message buffer.

As new tasks are dynamicallly created, the executive reserves the needed stack space. If no space is available, the executive deallocates a buffer from the System Buffer Pool and then allocates the needed stack space.

To send or receive a message, the executive allocates one or more buffers from the SBP for space to locate the message. With iDCX 51, messages can be optionally located in external (off-chip) memory. The pre-configured executive in the 8044 BEM BITBUS microcontroller, however, always locates messages in internal memory.

RELIABLE

Real-time control applications require reliability. The nucleus requires about 2.2K bytes of code space, 40 bytes on-chip RAM, and 218 bytes external RAM.

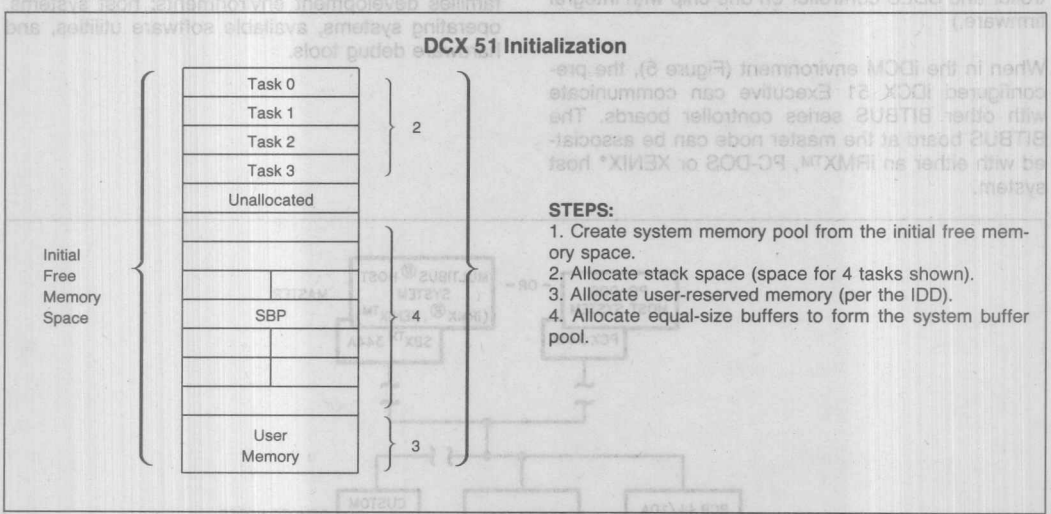


Figure 4. iDCX 51 Initialization of Internal Memory

Streamlined code increases performance and reliability, and flexibility is not sacrificed as code may be added to either on-chip or external memory.

The iDCX 51 architecture and simple user interface further enhance reliability and lower cost. For example, the straightforward structure of the user interfaces, and the transparent nature of the scheduling process contribute to reliability of the overall system by minimizing programming effort. Also, modularity increases reliability of the system and lowers cost by allowing user tasks to be refined independent of the system. In this way, errors are identified earlier and can be easily corrected in each isolated module.

In addition, users can assign tasks a Function ID that allows tracking of the tasks associated with a particular control/monitorig function. This feature reduces maintenance and trouble shooting time thus increasing system run time and decreasing cost.

OPERATING ENVIRONMENT

The iDCX 51 Executive supports applications development based on any member of the high performance 8051 family of microcontrollers. The Executive is available on diskette with user linkable libraries or in the 8044 BITBUS Enhanced Microcontroller preconfigured in on-chip ROM. (The 8044 BEM is an 8044 component that consists of an 8051 microcontroller and SDLC controller on one chip with integral firmware.)

When in the iDCM environment (Figure 5), the preconfigured iDCX 51 Executive can communicate with other BITBUS series controller boards. The BITBUS board at the master node can be associated with either an iRMX™, PC-DOS or XENIX* host system.

DEVELOPMENT ENVIRONMENT

Intel provides a complete development environment for the MCS-51 and RUPI-44 families of microcontrollers. The iDCX 51 Executive is only one of many of the software development products available. The executive is compatible with the following software development utilities available from Intel:

- 8051 Macro Assembler (ASM 51)
- PL/M 51 Compiler
- RL 51 Linker and Relocator Program
- LIB 51

Intel hardware development tools currently available for MCS-51 and RUPI-44 microcontroller development are:

- ICE-5100/252 Emulator for the MCS-51 family of microcontrollers
- ICE-5100/044 Emulator for the RUPI-44 family of microcontrollers (8044, 8344, 8744)
- iUP-200A/201A PROM Programmer, 21X software, and iUP programming modules

The DCX 51 Executive is also compatible with older hardware development tools (no longer available), which include:

- EMV-51/44 Emulation Vehicles
- ICE-51/44 In-Circuit Emulators

Table 4 shows the possible MCS-51 and RUPI-44 families development environments: host systems, operating systems, available software utilities, and hardware debug tools.

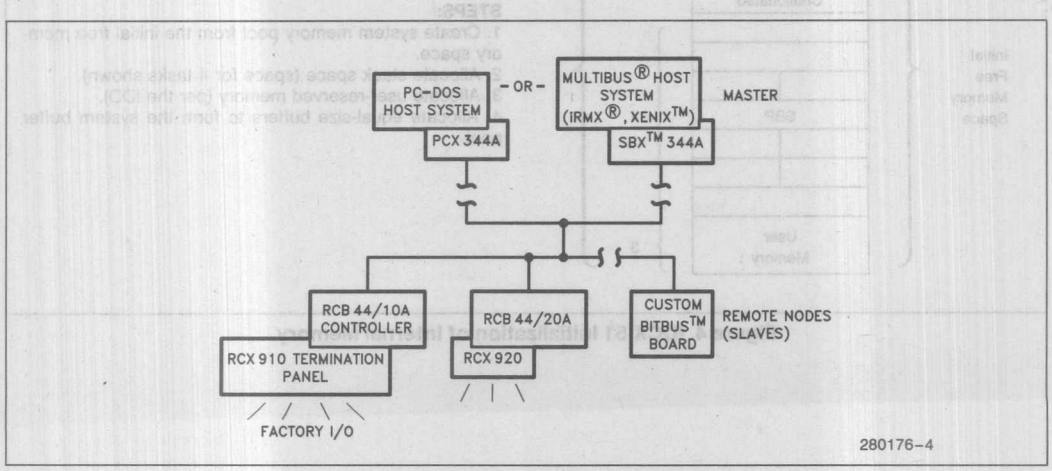


Figure 5. iDCM Operating Environment

SPECIFICATIONS

Supported Microcontrollers

8031	80C31
8051	80C51
8032	8751
8744	8044
8344	8052

Compatible DCM BITBUS™ Software

DCS 100 BITBUS Toolbox Host Software Utilities

DCS 110 BITWARE DCM44 Code for BITBUS emulation

Reference Manual (Supplied)

460367-001— iDCX 51 Distributes Control Executive User's Guide for Release 2.0.

ORDERING INFORMATION

Part Number	Description
DCX51SU	Executive for 8051 Family of Microcontrollers. Single User License, Development Only. Media Supplied for All Host Systems (Table 3).
DCX51RF	Royalty (Incorporation) Fee for iDCX Executive. Set of 50 incorporations. IDCX 51 RF does not ship with software (Order DCX 51SU).

Table 4. MCS®-51/RUPI™-44 Families Development Environments

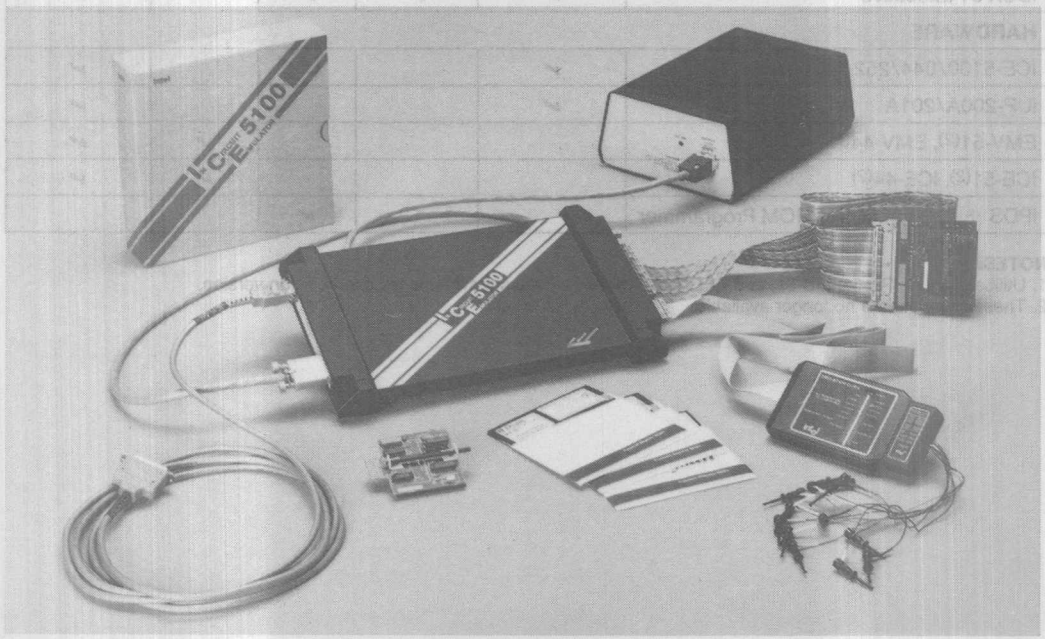
Development Utilities	Host Systems				
	PC/MS-DOS	iRMX® 86	iPDS™	Intellec®	
				Series II	Series III/IV
SOFTWARE					
ASM 51 + Utilities(1)	✓	✓	✓	✓	✓
PL/M 51 + Utilities(1)	✓	✓	✓	✓	✓
iDCX 51 Executive	✓	✓	✓		✓
HARDWARE					
ICE-5100/044/252	✓				✓
iUP-200A/201A	✓				✓
EMV-51(2), EMV-44(2)				✓	✓
ICE-51(2), ICE-44(2)				✓	✓
iPDS + iUP-F87/44A PROM Programmer			✓		

NOTES:
1. Utilities include RL 51, LIB 51, and AEDIT. Software for Series II systems is down-revision version.
2. These products are no longer available.

ICE™-5100/252
In-Circuit Emulator for the MCS®-51 Family of Microcontrollers

- **Precise, Full-Speed, Real-Time Emulation of Selected MCS-51 Microcontroller Components at Speeds Up to and Including 16 MHz**
- **64 KB of Mappable High-Speed Emulation Memory**
- **254 24-Bit Frames of Trace Memory (16 Bits Trace Program Execution Addresses and 8 Bits Trace External Events)**
- **Serial Link to the IBM* PC AT, PC XT (and DOS Compatibles), and the Intellec® Series III/IV**
- **ASM-51 and PL/M-51 Language Support**
- **Symbolic Debugging Enables Access to Memory Locations and Program Variables**
- **Four Address Breakpoints with In-Range, Out-Of-Range, and Page Breaks**
- **Equipped with the Integrated Command Directory (ICD™) that Includes:**
  - **— On-Line Help**
  - **— Syntax Guidance and Checking**
  - **— Dynamic Command-Entry**
  - **— Error Checking**
  - **— Command Recall**
- **On-Line Disassembler and Single-Line Assembler to Help with Code Patching**
- **Built-In CRT-Oriented Text Editor**

The ICE™-5100/252 in-circuit emulator is a high-level, interactive debugging system that is used to develop and test the hardware and software of a target system based on the MCS®-51 family of microcontrollers. The ICE-5100/252 emulator can be serially linked to an IBM PC AT or PC XT, or an Intellec Series III/IV. The emulator can communicate with the host system at standard baud rates up to 19.2K. The design of the emulator supports selected MCS-51 microcontroller components at speeds up to and including 16 MHz.



280200–1

*IBM is a registered trademark of International Business Machines Corporation.

November 1987
Order Number: 280200-002

PRODUCT OVERVIEW

The ICE-5100/252 emulator provides full emulation support for the MCS®-51 family members listed in Table 1.

The ICE-5100/252 emulator enables hardware and software development to proceed simultaneously. With the ICE-5100/252 emulator, prototype hardware can be added to the system as it is designed and software can be developed prior to the completion of the hardware prototype. Software and hardware integration can occur while the product is being developed.

The ICE-5100/252 emulator assists four stages of development:

- Software debugging
- Hardware debugging
- System integration
- System test

Software Debugging

The ICE-5100/252 emulator can be operated without being connected to the target system or before any of the user's hardware is available (provided external data RAM is not needed). In this stand-alone mode, the ICE-5100/252 emulator can be used to facilitate program development.

Hardware Debugging

The ICE-5100/252 emulator's AC/DC parametric characteristics match the microcontroller's. The emulator's full-speed operation makes it a valuable tool for debugging hardware, including time-critical serial port, timer, and external interrupt interfaces.

System Integration

Integration of software and hardware can begin when the emulator is plugged into the microcontroller socket of the prototype system hardware. Hardware can be added, modified, and tested immediately. As each section of the user's hardware is completed, it can be added to the prototype. Thus, the hardware and software can be system tested in real-time operation as each section becomes available.

System Test

When the prototype is complete, it is tested with the final version of the system software. The ICE-5100/252 emulator is then used for real-time emulation of the microcontroller to debug the system as a completed unit.

The final product verification test can be performed using the ROM or EPROM version of the microcontroller. Thus, the ICE-5100/252 emulator provides the ability to debug a prototype or production system at any stage in its development without introducing extraneous hardware or software test tools.

PHYSICAL DESCRIPTION

The ICE-5100/252 emulator consists of the following components (see Figure 1):

- Power supply
- AC and DC power cables
- Controller pod
- Serial cable (host-specific)
- User probe assembly (consisting of the processor module and the user cable)
- Crystal power accessory (CPA)

Table 1. MCS®-51 Family Support Offered by the ICE™-5100/252 Emulator

Part	On-Chip Program Memory	On-Chip Data Memory
8031	None	128 bytes
80C31	None	128 bytes
8032	None	256 bytes
8051	4 KB-ROM	128 bytes
80C51	4 KB-ROM	128 bytes
8052	8 KB-ROM	256 bytes
80C252	None	256 bytes
83C252	8 KB-ROM	256 bytes
8751	4 KB-EPROM	128 bytes
87C51	4 KB-EPROM	128 bytes
8752	8 KB-EPROM	256 bytes
87C252	8 KB-EPROM	256 bytes

- 40-pin DIP target adaptor
- Clips assembly
- Software (includes the ICE-5100/252 emulator software, diagnostic software, and tutorial)

The controller pod contains 64 KB of emulation memory, 254- by 24-bit frames of trace memory, and the control processor. In addition, the controller pod houses a BNC connector that can be used to connect up to 10 multi-ICE compatible emulators together for synchronous starting and stopping of emulation.

The serial cable connects the host system to the controller pod. The serial cable supports a subset of the RS-232C signals.

The user probe assembly consists of a user cable and a processor module. The processor module houses the emulation processor and the interface logic. The target adaptor connects to the processor module and provides an electrical and mechanical interface to the target microcontroller socket.

The crystal power accessory (CPA) is a small detachable board that connects to the controller pod and enables the ICE-5100/252 emulator to run in stand-alone mode. The target adaptor plugs into the socket on the CPA; the CPA then supplies clock and power to the user probe.

The clips assembly enables the user to trace external events. Eight bits of data are gathered on the rising edge of PSEN during opcode fetches. The clips information can be displayed using the CLIPS option with the PRINT command. Trace qualification input and output lines are also provided on the clips pod for connection to test equipment.

The ICE-5100/252 emulator software supports mnemonics, object file formats, and symbolic references generated by Intel's ASM-51 and PL/M-51 programming languages. Along with the ICE-5100/252 emulator software is a customer confidence test disk with diagnostic routines that check the operation of the hardware.

The on-line tutorial is written in the ICE-5100 command language. Thus, the user is able to interact with and use the ICE-5100/252 emulator while executing the tutorial.

A comprehensive set of documentation is included with the ICE-5100/252 emulator.

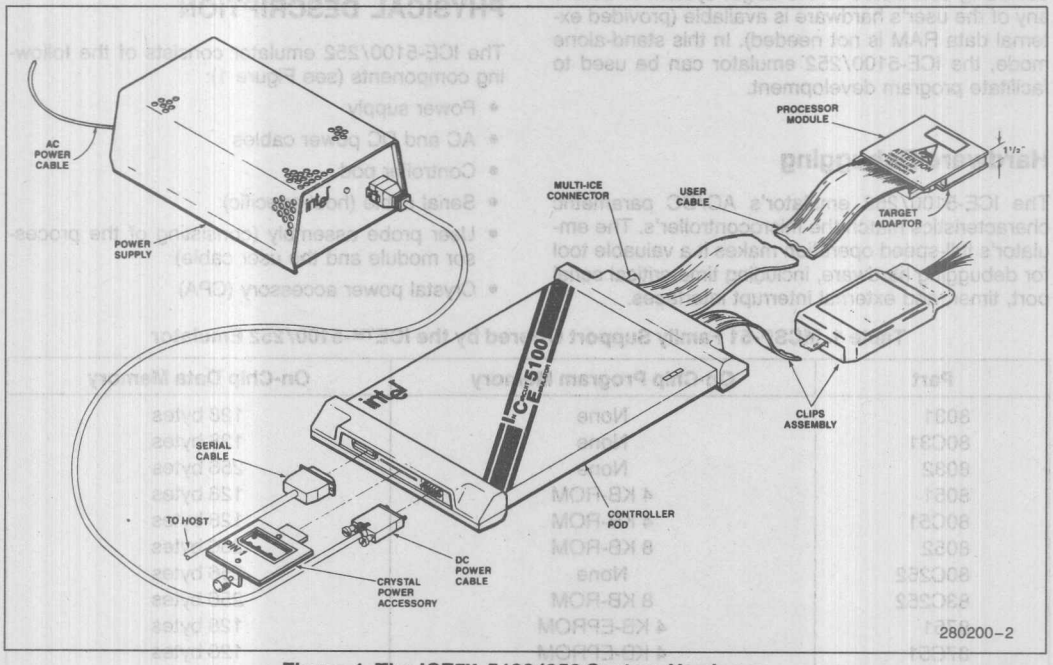


Figure 1. The ICE™-5100/252 System Hardware

ICE™-5100/252 EMULATOR FEATURES

The ICE-5100/252 emulator has been created to assist a product designer in developing, debugging, and testing designs incorporating the MCS®-51 family of microcontrollers. The following sections detail some of the ICE-5100/252 emulator features.

Processor Selection

The ICE-5100/252 emulator emulates the microcontrollers listed in Table 1. Selecting a processor type changes the following characteristics to match the microcontroller selected:

- Internal RAM size
- Internal ROM size
- Idle and power down mode enable
- Special function register symbolic map
- Memory map
- Latched or unlatched $\overline{EA}$
- Serial port framing and error detection

Emulation

Emulation is the controlled execution of the user's software in the target hardware or in an artificial hardware environment that duplicates the microcontroller of the target system. Emulation is a transparent process that happens in real-time. The execution of the user software is facilitated with the ICE-5100/252 command language.

Memory Mapping

There is 64 KB of memory that can be mapped to the CODE memory space in 4 KB blocks on 4 KB boundaries. By mapping memory to the ICE-5100/252 emulator, software development can proceed before the user hardware is available.

Memory Examination and Modification

The memory space for the MCS®-51 component(s) and its target hardware is fully accessible through the emulator. The ICE-5100/252 emulator refers to four physically distinct memory spaces, as follows:

- CODE — references program memory
- IDATA — references internal data memory
- RDATA — references special function register memory
- XDATA — references external data memory

ICE-5100/252 emulator commands that access memory use one of the special prefixes (e.g., CODE) to specify the memory space.

The microcontroller's special function registers and register bits can be accessed mnemonically (e.g., DPL, TCON, CY) with the ICE-5100/252 emulator software.

Data can be displayed or modified in one of three bases: hexadecimal, decimal, and binary. Data can also be displayed or modified in one of two formats: ASCII and unsigned integer. Program code can be disassembled and displayed as ASM-51 assembler mnemonics. Code can be modified with standard ASM-51 statements using the built-in single-line assembler.

Symbolic references can be used to specify memory locations. A symbolic reference is a procedure name, line number, program variable, or label in the user program that corresponds to a location.

Some typical symbolic functions include:

- Changing or inspecting the value of a program variable by using the symbolic name to access the memory location.
- Defining break and trace events using symbolic references.
- Referencing variables as primitive data types. The primitive data types are ADDRESS, BIT, BOOLEAN, BYTE, CHAR (character), and WORD.

The ICE-5100/252 emulator maintains a virtual symbol table (VST) for program symbols. A maximum of 61 KB of host memory space is available for the VST. If the VST is larger than 61 KB, the excess is stored on available host system disk space and is paged in and out as needed. The size of the VST is limited only by the disk capacity of the host system.

Breakpoint Specifications

Breakpoints are used to halt a user program in order to examine the effect of the program's execution on the target system. The ICE-5100/252 emulator supports three different types of break specifications:

- Specific address break — A single address can be specified to halt emulation.

- Range break — An arbitrary range of addresses can be specified to halt emulation. Program execution within or, optionally, outside the range halts emulation.
- Page break — Up to 256 page breaks can be specified. A page break is defined as a range of addresses that is 256-bytes long and begins on a 256-byte address boundary.

Break registers are user-defined debug definitions used to create and store breakpoint definitions. Break registers can contain multiple breakpoint definitions and can optionally call debug procedures when emulation halts.

Trace Specifications

Tracing can be triggered using specifications similar to those used for breaking. Normally, the ICE-5100/252 emulator traces program activity while the user program is executing. With a trace specification, tracing can be triggered to occur only when specific conditions are met during execution. Up to 254 24-bit frames of trace information are collected in the buffer during emulation. Sixteen of the 24 bits trace instruction execution addresses, and 8 bits capture external events (CLIPS).

The trace buffer display is similar to an ASM-51 program listing shown in Figure 2. The PRINT command enables the user to selectively display the contents of the trace buffer. The user has the option of displaying the clips information as well as disassembled instructions.

Procedures

Debugging procedures (PROCs) are a user-named group of ICE-5100/252 emulator commands that are executed as one command. PROCs enable the user to define several commands in a named block structure. The commands are executed by entering the name of the PROC. The PROC bodies are a simple DO...END construct.

PROCs can simulate missing hardware or software, collect debug information, and execute high-level software patches. PROCs can be copied to text files on disk, then recalled for use in later test sessions. PROCs can also serve as program diagnostics, implementing ICE-5100/252 emulator commands or user-defined definitions for special purposes. PROCs can also be used to set breakpoints.

On-Line Syntax Menu

A special menu, called the Integrated Command Directory (ICD), similar to the one used for the I²ICE™ system and the VLSiCE-96 emulator, aids in creating syntactically correct command lines. Figure 3 shows an example of the ICD and how it changes to reflect the options available for the GO command.

Help

The HELP command provides ICE-5100/252 emulation command assistance via the host system terminal. On-line HELP is available for the ICE-5100/252 emulator commands shown in Figure 4.

```
hlt>PRINT NEWEST 4      /* Print newest four instructions in
                           buffer */
FRAME       ADDRESS  CODE    INSTRUCTION
(028)       300A     C02A    PUSH  2AH
(030)       300C     2532    ADD   A,32H
(032)       300E     F52A    MOV   2AH,A
(034)       3010     B53210  CJNE  A,32H,$+10H
hlt>
hlt>PRINT CLIPS OLDEST 2  /* Buffer display showing clips */
FRAME       ADDRESS  CODE    INSTRUCTION      CLIPS (76543210)
(000)       300A     C02A    PUSH  2AH          01110011
(001)       300C     2532    ADD   A,32H        11110101
hlt>
```

280200-3

Figure 2. Selected Trace Buffer Displays

Design Considerations

The height of the processor module and the target adaptor need to be considered for target systems. Allow at least 1-½ inches (3.8 cm) of space to fit the processor module and target adaptor. Figure 5 shows the dimensions of the processor module.

```
hlt> GO
FROM  ARM  FOREVER   TIL  USING  TRACE  ;  <execute>

hlt> GO FROM
<expr>

hlt> GO FROM 13H
<operator> ARM FOREVER  TIL USING  TRACE ; <execute>

hlt> GO FROM 13H USING
BRKREG <brkreg name>

hlt> GO FROM 13H USING br1
; TRACE  ;  <execute>

hlt> GO FROM 13H USING br1 TRACE traceit
;  <execute>
```

280200-4

Figure 3. The Integrated Command Directory for the GO Command

```
hlt>HELP

HELP is available for:

ADDRESS    APPEND     ASM        BASE       BIT        BOOLEAN
BRKREG     BYTE       CHAR       CI         CNTL_C     COMMENTS
CONSTRUCTS COUNT      CPU        CURHOME    CURX       CURY
DCI        DEBUG      DEFINE     DIR        DISPLAY    DO
DYNASCOPE  EDIT       ERROR      EVAL       EXIT       EXPRESSION
GO         HELP       IF         INCLUDE    INVOCATION ISTEP
KEYS       LABEL      LINES      LIST       LITERALLY  LOAD
LSTEP      MAP        MENU       MODIFY     MODULE     MSPACE
MTYPE      NAMESCOPE  OPERATOR   PAGING     PARTITION  PRINT
PROC       PSEUDO_VAR PUT        REFERENCE  REGS       REMOVE
REPEAT     RESET      RETURN     SAVE       STRING     SYMBOLIC
SYNCSTART  TEMPCHECK  TRCREG     TYPES      VARIABLE   VERIFY
VERSION    WAIT       WORD       WRITE
hlt>
```

280200-5

Figure 4. HELP Menu

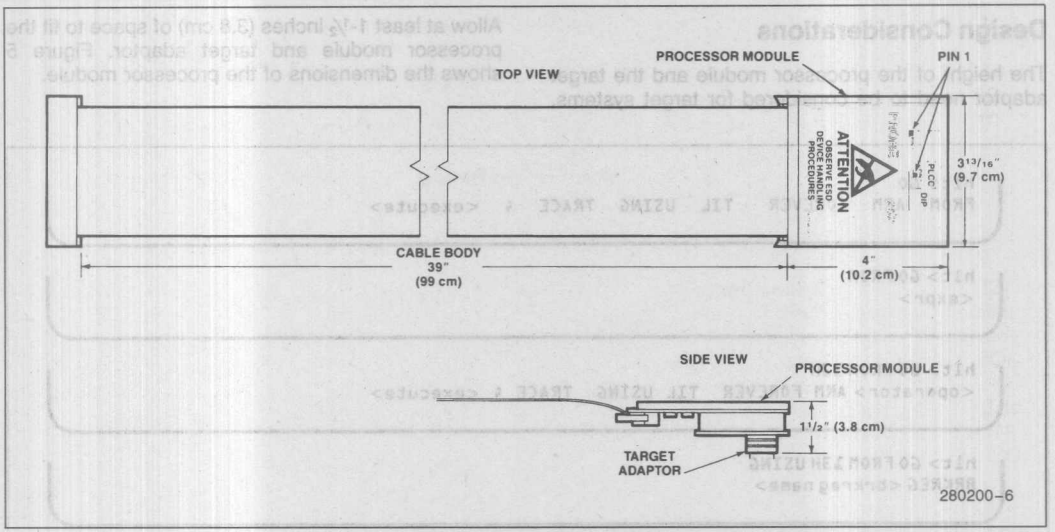


Figure 5. Processor Module Dimensions

ELECTRICAL CONSIDERATIONS

The emulation processor's user-pin timings and loadings are identical to the 80C252 component except as follows.

Maximum Operating ICC and Idle ICC (ma)*

V_{CC}	Maximum Operating ICC (ma)*			Maximum Idle ICC (ma)*		
	4V	5V	6V	4V	5V	6V
Frequency						
0.5 MHz	0.87	1.62	3.0	0.58	1.21	2.5
3.5 MHz	4.8	6.82	9.76	2.2	4.97	6.33
8.0 MHz	10.5	15.0	20.5	6.0	8.98	11.76
12.0 MHz	15.2	22.2	30.2	9.2	13.34	17.46
16.0 MHz	19.4	28.6	38.7	11.8	17.4	23.4

* ICC is measured with all output pins disconnected.
XTAL1 driven with TCLCH, TCHCL = 10ns, $V_{il} = V_{ss} + .5V$, $V_{ih} = Vcc - .5V$. XTAL2 not connected.
For maximum operating ICC
$\overline{EA}$ = RST = Port0 = Vcc.
For maximum idle ICC
* $\overline{EA}$ = Port0 = Vcc, RST = Vcc, internal clock to PCA gated off.

- Up to 25 pf of additional pin capacitance is contributed by the processor module and target adaptor assemblies.
- Pin 31, $\overline{EA}$, has approximately 32 pf of additional capacitance loading due to sensing circuitry.
- Pins 18 and 19, XTAL1 and XTAL2, respectively, have approximately 15 to 16 pf of additional capacitance when configured for crystal operation.

Chip Function	HMOS Component 8031	CHMOS Component 80C31
RST trigger threshold	2.5V	70% Vcc (3.5V @ Vcc = 5V)
RST input impedance	4K – 10K ohms	50K – 150K ohms
Port I_{IL}	$-800\mu A$	$-50\ \mu A$
Clock threshold	2.5V	70% Vcc (3.5V @ Vcc = 5V)

Emulating HMOS Components

The ICE-5100/252 emulator is based on a CHMOS emulation processor. There are minor differences between how the ICE-5100/252 emulator supports CHMOS and HMOS designs as shown in Table 2.

Refer to the *Microcontroller Handbook*, order number 210918, for further information on CHMOS and HMOS design considerations.

HOST REQUIREMENTS

- IBM PC AT or PC XT (or PC-DOS compatible) with 512 KB of RAM and a hard disk running under the DOS 3.0 (or later) operating system.
- Intellec Series III/IV Microcomputer Development System running under the ISIS or iNDX operating system respectively, with at least 512 KB of application memory resident.

  Disk drives — Dual floppy or one hard disk and one floppy drive required.

ICE™-5100/252 SYSTEM SOFTWARE PACKAGE

- ICE-5100/252 emulator software
- ICE-5100/252 confidence tests
- ICE-5100/252 tutorial software

EMULATOR PERFORMANCE

Memory

Mappable full-speed emulation code memory	64 KB	Mappable to user or ICE-5100/252 emulator memory in 4 KB blocks on 4 KB boundaries.
Trace Buffer		254- by 24- bit frames
Virtual Symbol Table		A maximum of 61 KB of host memory space is available for the Virtual Symbol Table (VST). The rest of the VST resides on disk and is paged in and out as needed.

PHYSICAL CHARACTERISTICS

Controller Pod

Width 8¼" (21 cm)
Height 1½" (3.8 cm)
Depth 13½" (34.3 cm)
Weight 4 lbs (1.85 kg)

User Cable

The user cable is 3 feet (approximately 1 m).

Processor Module

(with the target adaptor attached)

Width $3\frac{13}{16}$" (9.7 cm)
Length 1½" (3.8 cm)
Height 1½" (3.8 cm)

Power Supply

Width	7⅝" (18.1 cm)
Height	4" (10.06 cm)
Depth	11" (27.97 cm)
Weight	15 lbs (6.1 kg)

Serial Cable

The serial cable is 12 feet (3.6 m).

ELECTRICAL CHARACTERISTICS

Power Supply

100 - 120V or 200 - 240V (selectable)
50 - 60 Hz
2 amps (AC max) @ 120V
1 amp (AC max) @ 240V

ENVIRONMENTAL CHARACTERISTICS

Operating temperature	+10° C to +40°C (50°F to 104°F)
Operating humidity	Maximum of 85% relative humidity, non-condensing

ORDERING INFORMATION

Emulator Hardware and Software

Order Code	Description
pI252KITAD	This kit contains: ICE-5100/252 user probe assembly, power supply and cables, serial cables, target adaptor, CPA, ICE-5100 controller pod, software, and documentation for use with an IBM PC AT or PC XT. The kit also includes the 8051 Software Development Package and the AEDIT text editor for use on DOS systems. [Requires software license.]
pI252KITD	This kit is the same as the pI252KITAD kit excluding the 8051 Software Development Package and the AEDIT text editor. [Requires software license.]
pI252KITAS	This kit contains the ICE-5100/252 user probe assembly, power supply and cables, serial cables, target adaptor, CPA, ICE-5100 controller pod, software, and documentation for use with Intel hosts (Series III, IV). The kit also includes the 8051 Software Development Package and the AEDIT text editor for use on Series III/IV. [Requires software license.]
pI252KITS	This kit is the same as the pI252KITAS kit excluding the 8051 Software Development Package and the AEDIT text editor. [Requires software license.]

Software Only

Order Code	Description
pSA252D	This kit contains the host, probe, diagnostic and tutorial software on 5¼" disks for use on an IBM PC AT or PC XT (requires DOS 3.0 or later). [Requires software license.]
pSA252S	This kit contains the host, probe, diagnostic and tutorial software on 8" disks (both single-density and double-density) for use on a Series III, and on 5¼" disks for use on a Series IV. [Requires software license.]

Other Useful Intel Debug and Development Support Products

Order Code	Description
pD86ASM51	8051 Software Development Package (DOS version) — Consists of the ASM-51 macro assembler which gives symbolic access to 8051 hardware features; the RL51 linker and relocator program that links modules generated by ASM-51; CONV51 which enables software written for the MCS-48 family to be up-graded to run on the 8051, and the LIB51 Librarian which programmers can use to create and maintain libraries of software object modules. Use with the DOS operating system (version 3.0 or later).

pD86PLM51	PL/M-51 Software Package (DOS version) — Consists of the PL/M-51 compiler which provides high-level programming language support; the LIB51 utility that creates and maintains libraries of software object modules, and the RL51 linker and relocator program that links modules generated by ASM-51 and PL/M-51 and locates the linked object modules to absolute memory locations. Use the DOS operating system (version 3.0 or later).
pI86ASM51	8051 Software Development Package (ISIS version) — Same as the pD86ASM51 package except this one is for use with the Series III.
pI86PLM51	PL/M-51 Software Package — Same as the pD86PLM51 package except this one is for use with the Series III and Series IV.
pD86EDIEU	AEDIT text editor for use with the DOS operating system.

MCS®-51 INDEX

8
8031AH, 5-2
8032AH, 5-2
8051, 5-2
8051AH, 5-2
8052AH, 5-2
8052AH-BASIC, 5-2
80C31BH, 5-2
80C51BH, 5-2, 6-28
80C51FA, 5-2, 7-1
83C51FA, 5-2, 7-1
80C152, 5-3
83C152, 5-3
8751H, 5-2, 6-25, 6-26
8752BH, 5-2, 6-26
87C51, 5-2, 6-26, 6-27
87C51FA, 5-2, 7-1

A
AC (Auxiliary Carry) Flag, 6-3, 9-10
ACALL, 5-13, 9-24
Accumulator, 6-2
ADD, 9-25, 9-35
ADDC, 9-26, 9-35
ADDRESS/DATA Bus, 6-4, 6-5
Addressing
  Broadcast, 7-8
  Given, 7-8
AJMP, 5-13, 9-28
ALE, 5-16, 6-6, 6-30
ANL, 5-12, 6-6, 9-28
Arithmetic Instructions, 5-7
ASM51, 5-6
Automatic Address Recognition, 7-8

B
B Register, 6-2
Baud Rate, 6-12, 6-13, 6-14, 6-17
BCD, 5-8, 5-10
Bit Addressable, 9-5
Boolean Instructions, 5-11, 5-12
Bus Cycle
  Data Memory, 5-16
  Program Memory, 5-16
Byte Addressable, 9-7

C
C (Carry) Flag, 5-6, 5-12, 6-3, 9-10
Capture Mode, 7-4
Capture Registers, 6-2
Case Jump, 5-7, 5-13
Ceramic Resonator, 5-14, 6-28
CJNE, 5-14, 9-31
CLR, 5-12, 6-6, 9-33
Compare Mode, 7-6
Control Registers, 6-2
CPL, 5-12, 6-6, 9-34
CPU Timing, 5-14
Crystal, 5-14, 6-27, 6-28

D
DA A, 5-8, 9-35
Data Memory, 5-4, 5-5, 5-11, 6-6, 6-30, 9-3
  Read Cycle, 6-31
  Write Cycle, 6-31
Data Pointer, 6-2
Data Transfers, 5-9
DEC, 6-6, 9-37
Direct Addressing, 5-7, 9-3, 9-5
DIV AB, 5-8, 6-22, 9-38
DJNZ, 5-14, 6-6, 9-39

E
EA (External Access), 5-4, 6-6, 6-21, 6-26, 6-30
Encryption Array, 6-26
EPROM Programming, 6-26, 6-30
EPROM Verifying, 6-26
Execution Times, 5-7
External Clock, 6-28, 6-29
External Program Execution, 5-4

F
Fetch/Execution Sequence, 5-15
Framing Error Detection, 7-8

H
High Speed Output, 7-4, 7-7

I
I/O Buffers, 6-3, 6-4
Idle Mode, 5-2, 6-24, 6-25, 7-9, 9-10
IE (Interrupt Enable), 5-17, 6-2, 6-20, 7-13, 9-11
Immediate Constants, 5-7
INC, 6-6, 9-40
Indexed Addressing, 5-7
Indirect Addressing, 5-7, 9-3, 9-5
Instruction Set, 5-6, 9-20
Interrupt Response Time, 6-21, 6-22
Interrupts, 5-4, 5-17, 6-20, 7-11, 9-11
  External, 6-22, 9-11
IP (Interrupt Priority), 5-17, 6-2, 6-21, 7-13, 9-12

J
JB, 5-12, 9-42
JBC, 5-12, 6-6, 9-42
JC, 5-12, 9-43
JMP, 5-13, 9-44

JNZ, 9-46
Jump Instructions, 5-13
JZ, 9-46

L
LCALL, 5-13, 9-47
LJMP, 5-13, 9-47
Lock Bits, 6-26
Logical Instructions, 5-9
Lookup Tables, 5-7, 5-11

M
Machine Cycle, 5-15
MOV, 5-12, 6-6, 9-48
  16-Bit, 5-10
MOVC, 5-11, 9-53
MOVX, 5-11, 5-16, 9-54
MUL AB, 5-7, 6-22, 9-56
Multiprocessor Communication, 6-11

N
Ninth Data Bit, 6-11, 6-17
NOP, 9-56

O
ONCE (On-Circuit Emulation) Mode, 6-27
ORL, 5-12, 6-6, 9-57
Oscillator, 5-14, 6-23, 6-27
  External, 5-15
Oscillator Frequency, 6-12
OV (Overflow) Flag, 6-3, 9-10

P
P (Parity) Flag, 5-7, 6-3, 9-10
PCA Timer/Counter, 7-1, 7-5
PCON, 6-2, 6-12, 6-24, 9-10, 9-19
Polling Sequence, 5-17
POP, 5-9, 9-60
Port Bit Latch, 6-3, 6-4
Ports, 6-2, 6-3, 6-29
Power Down Mode, 5-2, 6-24, 6-25, 7-9, 9-10
Power Off Flag, 7-9
PROG, 6-30
Program Memory, 5-3, 5-4, 6-6, 6-26, 9-2
Program Memory Locks, 6-26
Programmable Counter Array (PCA), 7-1
PSEN, 5-4, 5-16, 6-6, 6-30
PSW (Program Status Word), 5-6, 6-2, 9-10
Pulse Width Modulator (PWM), 7-4, 7-7
PUSH, 5-9, 9-60

Q
Quick-Pulse Programming Algorithm, 6-26

Register Banks, 5-4, 6-3, 9-5, 9-10
Register Instructions, 5-7
Register-Specific Instructions, 5-7
Relative Offset, 5-12
Reset, 6-23, 6-30, 9-8
  Power-On, 6-24
RET, 5-13, 6-22, 9-61
RETI, 5-13, 6-21, 6-22, 9-61
RI (Receive Interrupt) Flag, 6-12, 6-14, 6-17, 6-20, 9-18
RL, 9-62
RLC, 9-62
RR, 9-63
RRC, 9-63

S
SBUF, 6-2, 6-10, 6-14
SCON, 6-2, 6-11, 6-14, 6-17, 9-18
Serial Port, 6-10, 6-15, 7-8, 9-10, 9-12, 9-16, 9-18, 9-19
SET, 6-6
SETB, 5-12, 9-64
SFRs, 5-6, 6-1, 6-2, 6-4, 7-15, 9-4, 9-5, 9-7
SJMP, 5-13, 9-65
Software Timer, 7-4, 7-6
Stack Pointer, 6-2, 9-5
Start Bit, 6-11, 6-14, 6-17
State Time, 5-15
Status Flags, 5-7
Stop Bit, 6-11, 6-14, 6-17
SUBB, 9-66
SWAP A, 5-9, 9-67

T
T2CON, 6-2, 6-9, 6-10, 6-13, 9-16, 9-17
TCON, 6-2, 6-7, 6-8, 6-22, 9-11, 9-13
Third Priority Level, 5-18
TI (Transmit Interrupt) Flag, 6-12, 6-14, 6-17, 6-20, 9-18
Timer/Counters, 6-2, 6-6, 9-13
  Up/Down, 7-9
TMOD, 6-2, 6-7, 9-13, 9-14, 9-17

V
V_{PP}, 6-26, 6-30

W
Watch Dog Timer, 7-3, 7-8
WR Signal, 5-4, 6-6

X
XCH, 5-10, 9-68
XCHD, 5-10, 9-69
XRL, 5-12, 6-6, 9-69
XTAL1, 5-14, 5-15, 6-28, 6-29, 6-30
XTAL2, 5-14, 5-15, 6-28, 6-29, 6-30

80C152 INDEX

A
A, 8-4, 8-6
Abort, 8-29, 8-30
Acknowledgement, 8-15, 8-16, 8-24, 8-25, 8-30
Acquisition Time, 8-21
Address, 8-18, 8-25, 8-30, 8-35
Address Assignment, 8-34, 8-35
Address Length
  see AL
Address Mask Registers
  see AMSKn
Address Match Registers
  see ADRn
Address Negotiation, 8-34
Address Recognition, 8-15, 8-16, 8-18
ADR0, 8-3, 8-4, 8-6, 8-41
ADR1, 8-3, 8-4, 8-6, 8-41
ADR2, 8-3, 8-4, 8-6, 8-41
ADR3, 8-3, 8-4, 8-6, 8-41
AE, 8-3, 8-9, 8-43
AL, 8-3, 8-18, 8-41
ALE, 8-13, 8-49
ALE Switch, 8-49
Alignment Error
  see AE
Alternate Backoff, 8-15, 8-16, 8-19, 8-21, 8-22, 8-23, 8-40
Alternate Cycle Mode, 8-46, 8-52
AMSK0, 8-3, 8-4, 8-6, 8-41
AMSK1, 8-3, 8-4, 8-6, 8-41
ARB, 8-49, 8-52
Arbiter
  see ARB
Arbiter Mode, 8-47, 8-48, 8-49

B
B, 8-4, 8-6
Backoff Algorithm, 8-19, 8-20, 8-39, 8-40
Backoff Mode
  see M0, M1
Backoff Timer
  see BKOFF
Back-to-Back Frames, 8-42
Balanced, 8-30
Baud, 8-3, 8-4, 8-6, 8-20, 8-41
Baud Rate, 8-17, 8-34
Baud Rate Generator
  see Baud
BCRH0, 8-3, 8-4, 8-6, 8-45, 8-46
BCRH1, 8-3, 8-4, 8-6, 8-45, 8-46
BCRL0, 8-3, 8-4, 8-6, 8-45, 8-46
BCRL1, 8-3, 8-4, 8-6, 8-45, 8-46
Beginning of Frame Flag
  see BOF
Bit Addressable Memory, 8-6
Bit Addresses, 8-7
Bit Addresses (Symbolic), 8-8
Bit Rate, 8-17
Bit Stripping, 8-25, 8-29, 8-37
Bit Stuffing, 8-25, 8-29, 8-35, 8-37
Bit Time, 8-21
BKOFF, 8-3, 8-4, 8-6, 8-41
Block Diagram, 8-2
BOF, 8-17, 8-18, 8-20, 8-25, 8-37, 8-39
Broadcast Address, 8-18, 8-25
Burst Mode, 8-46, 8-51, 8-52
Byte Count, 8-33

C
Clock Recovery, 8-17, 8-37, 8-38
Collision, 8-17, 8-20, 8-39
Collision Fragment, 8-21
Collision Resolution, 8-15, 8-16, 8-19, 8-21
Command, 8-26
Control Field, 8-25, 8-26, 8-28, 8-30
CRC, 8-3, 8-15, 8-16, 8-18, 8-21, 8-24, 8-25, 8-28, 8-32, 8-35, 8-37, 8-44
CRC Error
  see CRCE
CRC Generating Polynomial, 8-18, 8-28
CRC Jam, 8-15, 8-16
CRC Remainder, 8-18, 8-28
CRC Type
  see CT
CRCE, 8-3, 8-9, 8-43
Crystal Selection, 8-17
CSMA/CD, 8-14, 8-15, 8-16, 8-17, 8-34, 8-35, 8-37, 8-40
CSMA/CD Frame
  see Frame Format
CT, 8-3, 8-41
Cyclic Redundancy Check
  see CRC

D
DARH0, 8-3, 8-4, 8-6, 8-45
DARH1, 8-3, 8-4, 8-6, 8-45
DARL0, 8-3, 8-4, 8-6, 8-45
DARL1, 8-3, 8-4, 8-6, 8-45
DAS, 8-3, 8-45, 8-52
Data Encoding, 8-15, 8-16, 8-29, 8-34
Data Memory, 8-5
DC Jam, 8-15, 8-16, 8-21
DC Jam (Bit)
  see DCJ
DCJ, 8-3, 8-21, 8-42
DCON0, 8-3, 8-4, 8-6, 8-33, 8-45, 8-46, 8-52
DCON1, 8-3, 8-4, 8-6, 8-33, 8-45, 8-46, 8-52
DCR, 8-3, 8-42

D (Continued)
Deference, 8-17
Demand Mode, 8-46, 8-51, 8-52
Demand Mode (Bit)
see DM
DEN, 8-11, 8-13, 8-29, 8-35, 8-44
Destination Address Space
see DAS
Deterministic Backoff, 8-15, 8-16, 8-17, 8-19, 8-21, 8-22, 8-23, 8-40, 8-43
Deterministic Collision Resolution (Bit)
see DCR
Direct Memory Access
see DMA
DM, 8-3, 8-46, 8-52
DMA, 8-24
DMA Arbitration, 8-51
DMA Control Bits, 8-52
DMA Control Register
see DCONn
DMA Cycle, 8-45, 8-46, 8-47, 8-51
DMA Precedence, 8-51
DMA Register, 8-45
DMA Register Access, 8-51
DMA Select
see DMA (Bit)
DMA Serial Demand Mode
see GSC, Servicing of
DMA Timing, 8-47
DMA Transfer, 8-44, 8-45, 8-47, 8-48
DMA (Bit), 8-9, 8-33, 8-43
DMA0, 8-44
DMA1, 8-44
DONE, 8-3, 8-9, 8-33, 8-46, 8-52
DPH, 8-4, 8-6
DPL, 8-4, 8-6
DPTR, 8-6
Duplex
see Full Duplex

E
EA, 8-13
EDMA0, 8-9, 8-10
EDMA1, 8-9, 8-10
EGSRE, 8-9, 8-10, 8-20, 8-33
EGSRV, 8-9, 8-10, 8-33
EGSTE, 8-9, 8-10, 8-20, 8-33
EGSTV, 8-9, 8-10, 8-33
End of Frame Flag
see EOF
Ending Reception, 8-40
Ending Transmission, 8-40
EOF, 8-17, 8-18, 8-25, 8-29, 8-40
Error Reporting, 8-25
ES, 8-10
ET0, 8-10
ET1, 8-10
EX0, 8-10
EX1, 8-10
Extended Address, 8-18
External Clocking of GSC, 8-15, 8-16, 8-17, 8-37
External Demand Mode, 8-46
External Driver, 8-35
External Transmit Clock
see XTCLK

F
FIFO Pointer, 8-43
Flags, 8-15, 8-16, 8-35
Frame Format, 8-17, 8-25
Full Duplex, 8-15, 8-16, 8-24, 8-30, 8-32, 8-34, 8-37

G
Garen, 8-42
GF0, 8-1, 8-5
GF1, 8-1, 8-5
GFIEN, 8-1, 8-5, 8-42
Global Serial Channel
see GSC
Glossary, 8-52
GMOD, 8-3, 8-4, 8-6, 8-41
GO, 8-3, 8-46, 8-52
GREN, 8-3, 8-24, 8-39, 8-40, 8-43
Group Address
see Multicast Address
GRXD, 8-13
GSC, 8-14, 8-46
GSC
Servicing of (CPU), 8-32, 8-39
Servicing of (DMA), 8-15, 8-16, 8-32, 8-33, 8-39
Servicing of (GSC), 8-15, 8-16
GSC Auxiliary Receiver Enable
see GAREN
GSC External Receive Clock Enable
see XRCLK
GSC Idle Flag Enable
see GFIEN
GSC Mode
see GMOD
GSC Operation, 8-37
GSC Receive Enable
see GREN
GSC Register Descriptions, 8-41
GSC Sampling Rate, 8-20, 8-37
GSC Transmit FIFO
see TFIFO
GTXD, 8-13

H
HABEN, 8-3, 8-24, 8-39, 8-43
Half Duplex, 8-15, 8-16, 8-24, 8-30, 8-32, 8-34, 8-37
Hardware Based Acknowledge Enable
see HABEN
Hardware Based Acknowledgement
see HBA
HBA, 8-15, 8-16, 8-24, 8-30
HDLC, 8-30, 8-37
$\overline{\text{HLDA}}$, 8-13, 8-44, 8-47, 8-48, 8-49, 8-52
$\overline{\text{HOLD}}$, 8-13, 8-44, 8-47, 8-48, 8-49, 8-52
Hold Acknowledge
see HLDA
Hold Request
see HOLD

I
IDA, 8-3, 8-45, 8-52
Idle, 8-14, 8-40
Idle Fill Flags, 8-42
Idle (GSC), 8-30
IE, 8-4, 8-6, 8-10
IEN1, 8-3, 8-4, 8-6, 8-9, 8-10
IFS, 8-3, 8-4, 8-6, 8-19, 8-29, 8-41, 8-42
Increment Destination Address
see IDA
Increment Source Address
see ISA
Information Field, 8-18, 8-25, 8-28
Information Frame, 8-25, 8-26
Initialization (GSC), 8-34, 8-35
$\overline{\text{INT0}}$, 8-13, 8-46
$\overline{\text{INT1}}$, 8-13, 8-46
Interframe Space, 8-17, 8-19, 8-22, 8-24, 8-32, 8-40
Interframe Space (Register)
see IFS
Internal Clocking of GSC, 8-15, 8-16
Interrupt Structure, 8-9
IP, 8-4, 8-6, 8-10
IPN1, 8-3, 8-4, 8-6, 8-9, 8-10
ISA, 8-45, 8-52

J
Jam, 8-15, 8-16, 8-17, 8-21, 8-22, 8-40
Jam Time, 8-21
Jitter, 8-35, 8-36

L
Line Discipline, 8-30, 8-34, 8-37
Line Idle
see LNI
LNI, 8-5, 8-30, 8-44
Local Serial Channel
see LSC
Loop Configuration, 8-30
Loopback, 8-35
LSC, 8-14, 8-46

M
M0, 8-3, 8-35, 8-40, 8-41
M1, 8-3, 8-35, 8-40, 8-41
Manchester, 8-15, 8-16, 8-17, 8-20, 8-24, 8-35, 8-37
Master Station
see Primary Station
Memory Space, 8-1
Misalignment, 8-24
Mode, 8-26
Modulo, 8, 8-30
Modulo, 128, 8-30
Monitoring Link, 8-40
MOVX, 8-49
Multi-Drop Configuration, 8-30, 8-31
Myslot, 8-3, 8-4, 8-6, 8-22, 8-42

N
Network Changes, 8-32
Network Expansion, 8-32
No Acknowledgement
see NOACK
NOACK, 8-5, 8-9, 8-24, 8-44
Nonsequenced Frame
see Unnumbered Frame
Normal Backoff, 8-15, 8-16, 8-19, 8-21, 8-22, 8-23, 8-40
NRZ, 8-15, 8-16, 8-17, 8-24, 8-35, 8-37
NRZI, 8-15, 8-16, 8-29, 8-35, 8-37
Number of Stations, 8-40

O
Overflow, 8-24
Overrun
see OVR
OVR, 8-3, 8-9, 8-43

P
P0
see PORT0
P1
see PORT1
P2
see PORT2
P3
see PORT3
P4
see PORT4
Package, 8-11
PCON, 8-4, 8-6, 8-42, 8-48, 8-49, 8-52
PDMA0, 8-9, 8-10
PDMA1, 8-9, 8-10
PGSRE, 8-9, 8-10
PGSRV, 8-9, 8-10
PGSTE, 8-9, 8-10
PGSTV, 8-9, 8-10
Pin Description, 8-12
Pin Out (DIP), 8-11
Pin Out (PLCC), 8-12
PL0, 8-3, 8-41
PL1, 8-3, 8-41

P (Continued)
Point-to-Point Configuration, 8-30, 8-31
Poll/Final Bit
see P/F Bit
PORT0, 8-4, 8-6, 8-12, 8-44
PORT1, 8-4, 8-6, 8-12
PORT2, 8-4, 8-6, 8-13, 8-44
PORT3, 8-4, 8-6, 8-13
PORT4, 8-3, 8-4, 8-6, 8-11, 8-13
Power Down, 8-14
PR, 8-3, 8-41
PRBS, 8-3, 8-4, 8-6, 8-22, 8-42
Preamble, 8-15, 8-16, 8-17, 8-18, 8-24, 8-35, 8-37
Preamble Length
see PL
Primary Station, 8-24, 8-25, 8-30
Program Memory, 8-8
Program Verification, 8-12, 8-13
Promiscuous Address, 8-25
Protocol (Bit)
see PR
PS, 8-10
$\overline{\text{PSEN}}$, 8-13
Pseudo Random Binary Sequence
see PRBS
PSW, 8-4, 8-6
PT0, 8-10
PT1, 8-10
PX0, 8-10
PX1, 8-10
P/F Bit, 8-25, 8-26

R
Raw Receive, 8-15, 8-16, 8-17, 8-35
Raw Transmit, 8-15, 8-16, 8-17, 8-29, 8-35
RCABT, 8-3, 8-9, 8-20, 8-29, 8-39, 8-40, 8-43
$\overline{\text{RD}}$, 8-13, 8-44, 8-47, 8-51
RDN, 8-3, 8-9, 8-33, 8-39, 8-40, 8-43
Receive Count, 8-30
Receive FIFO
see RFIFO
Receive FIFO Not Ready
see RFNE
Receive Status Register
see RSTAT
Receiver Collision/Abort Detect
see RCABT
Receiver Done
see RDN
Reception Sequence, 8-26
REN, 8-20, 8-40
REQ, 8-49, 8-52
Requester Mode, 8-47, 8-48, 8-49
Requester (Bit)
see REQ
Reset, 8-1, 8-10, 8-11, 8-13, 8-14, 8-19, 8-42
Resolution Phase, 8-17, 8-21, 8-23
Response, 8-26
Retransmission, 8-39
RFIFO, 8-3, 8-4, 8-6, 8-20, 8-32, 8-33, 8-39, 8-40, 8-43, 8-46
RFIFO Time Delay, 8-20
RFNE, 8-3, 8-9, 8-33, 8-39, 8-43, 8-46
RI, 8-46
Ring Configuration, 8-30, 8-31
Round-Trip Propagation, 8-21
RSTAT, 8-3, 8-4, 8-6, 8-43
$\overline{\text{RXC}}$, 8-37
RXD, 8-13

S
SARH0, 8-3, 8-4, 8-6, 8-33, 8-45
SARH1, 8-3, 8-4, 8-6, 8-45
SARL0, 8-3, 8-4, 8-6, 8-33, 8-45
SARL1, 8-3, 8-4, 8-6, 8-45
SAS, 8-3, 8-45, 8-52
SBUF, 8-4, 8-6, 8-46
SCON, 8-4, 8-6
SDLC, 8-14, 8-15, 8-16, 8-24, 8-30, 8-34, 8-35, 8-37, 8-40
SDLC Commands, 8-27
SDLC Frame
see Frame Format
Secondary Station, 8-24, 8-25, 8-30
Sending Sequence, 8-26
Separation of Busses, 8-50
Sequence Count, 8-25, 8-30
Serial Backplane, 8-44
Serial Port Demand Mode, 8-46
SFRS
see Special Function Registers
Slave Station
see Secondary Station
Slot Address, 8-42
Slot Address Register
see Myslot
Slot Assignment, 8-17
Slot Time, 8-21, 8-22, 8-32, 8-40
Slot Time (Register)
see SLOTTM
SLOTTM, 8-3, 8-4, 8-6, 8-21, 8-43
Source Address Space
see SAS
SP, 8-4, 8-6
Special Function Registers, 8-3, 8-6
Supervisory Frame, 8-25, 8-26

T
T0, 8-13
T1, 8-13
TCDCNT, 8-3, 8-4, 8-6, 8-22, 8-39, 8-40, 8-43
TCDT, 8-5, 8-9, 8-20, 8-39, 8-40, 8-44
TCON, 8-4, 8-6
TDN, 8-5, 8-9, 8-24, 8-33, 8-39, 8-40, 8-43, 8-44
TEN, 8-5, 8-29, 8-39, 8-40, 8-43, 8-44
Test Modes, 8-35
TFIFO, 8-4, 8-5, 8-6, 8-29, 8-32, 8-33, 8-39, 8-40, 8-43, 8-44, 8-46
TFNF, 8-5, 8-9, 8-39, 8-44, 8-46
TH0, 8-4, 8-6
TH1, 8-4, 8-6

T (Continued)
TI, 8-46
Timer/Counters, 8-11
TL0, 8-4, 8-6
TL1, 8-4, 8-6
TM, 8-3, 8-46, 8-52
TMOD, 8-4, 8-6
Transfer Mode
  see TM
Transmission During Resolution, 8-40
Transmit Collision Detect
  see TCDT
Transmit Collision Detect Count
  see TCDCNT
Transmit Done
  see TDN
Transmit Enable
  see TEN
Transmit FIFO Not Full
  see TFNF
Transmit Status Register
  see TSTAT
Transmit Waveforms, 8-37
TSTAT, 8-4, 8-5, 8-6, 8-43
Turn-Around Time, 8-19
$\overline{\text{TXC}}$, 8-3, 8-37
TXD, 8-13

U
UART
  see LSC
Unbalanced, 8-30
Underrun
  see UR
Unnumbered Frame, 8-25, 8-26, 8-27, 8-28
Unused SFR Addresses, 8-5
UR, 8-5, 8-9, 8-43, 8-44
User Defined Protocols, 8-30
Using GSC, 8-30
Using HLDA, 8-49
Using HOLD, 8-, 8-49

V
V_{CC}, 8-12
V_{SS}, 8-12

W
$\overline{\text{WR}}$, 8-13, 8-44, 8-47, 8-51

X
XRCLK, 8-1, 8-5, 8-37, 8-42
XTAL1, 8-12
XTAL2, 8-12
XTCLK, 8-37, 8-41

THE RUPI™-44 FAMILY: MICROCONTROLLER WITH ON-CHIP COMMUNICATION CONTROLLER

INTRODUCTION

The RUPI-44 family is designed for applications requiring local intelligence at remote nodes, and communication capability among these distributed nodes. The RUPI-44 integrates onto a single chip Intel's highest performance microcontroller, the 8051-core, with an intelligent and high performance Serial communication controller, called the Serial Interface Unit, or SIU. See Figure 1. This dual controller architecture allows complex control and high speed data communication functions to be realized cost effectively.

The RUPI-44 family consists of three pin compatible parts:

- 8344—8051 Microcontroller with SIU
- 8044—An 8344 with 4K bytes of on-chip ROM program memory
- 8744—An 8344 with 4K bytes of on-chip EPROM program memory

1.0 ARCHITECTURE OVERVIEW

The 8044's dual controller architecture enables the RUPI to perform complex control tasks and high speed communication in a distributed network environment.

The 8044 microcontroller is the 8051-core, and maintains complete software compatibility with it. The microcontroller contains a powerful CPU with on-chip peripherals, making it capable of serving sophisticated real-time control applications such as instrumentation, industrial control, and intelligent computer peripherals. The microcontroller features on-chip peripherals such as two 16-bit timer/counters and 5 source interrupt capability with programmable priority levels. The microcontroller's high performance CPU executes most instructions in 1 microsecond, and can perform an 8×8 multiply in 4 microseconds. The CPU features a Boolean processor that can perform operations on 256 directly addressable bits. 192 bytes of on-chip data RAM can be extended to 64K bytes externally. 4K bytes of on-chip program ROM can be extended to 64K bytes externally. The CPU and SIU run concurrently. See Figure 2.

The SIU is designed to perform serial communications with little or no CPU involvement. The SIU supports data rates up to 2.4 Mbps, externally clocked, and 375 Kbps self clocked (i.e., the data clock is recovered by an on-chip digital phase locked loop). SIU hardware supports the HDLC/SDLC protocol: zero bit insertion/deletion, address recognition, cyclic redundancy check, and frame number sequence check are automatically performed.

The SIU's Auto mode greatly reduces communication software overhead. The AUTO mode supports the SDLC Normal Response Mode, by performing secondary station responses in hardware without any CPU involvement. The Auto mode's interrupt control and frame sequence numbering capability eliminates software overhead normally required in conventional systems. By using the Auto mode, the CPU is free to concentrate on real time control of the application.

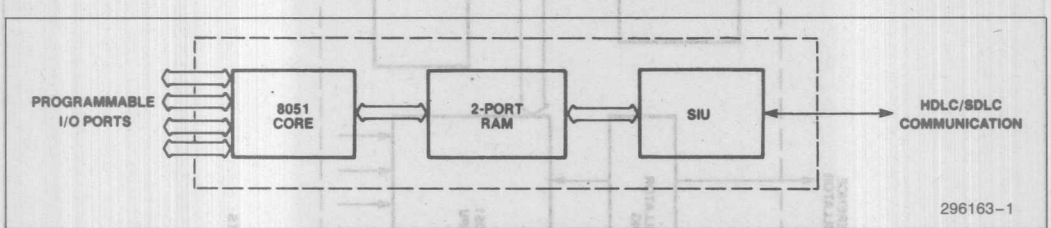


Figure 1. RUPI™-44 Dual Controller Architecture

Order Number: 296163-001

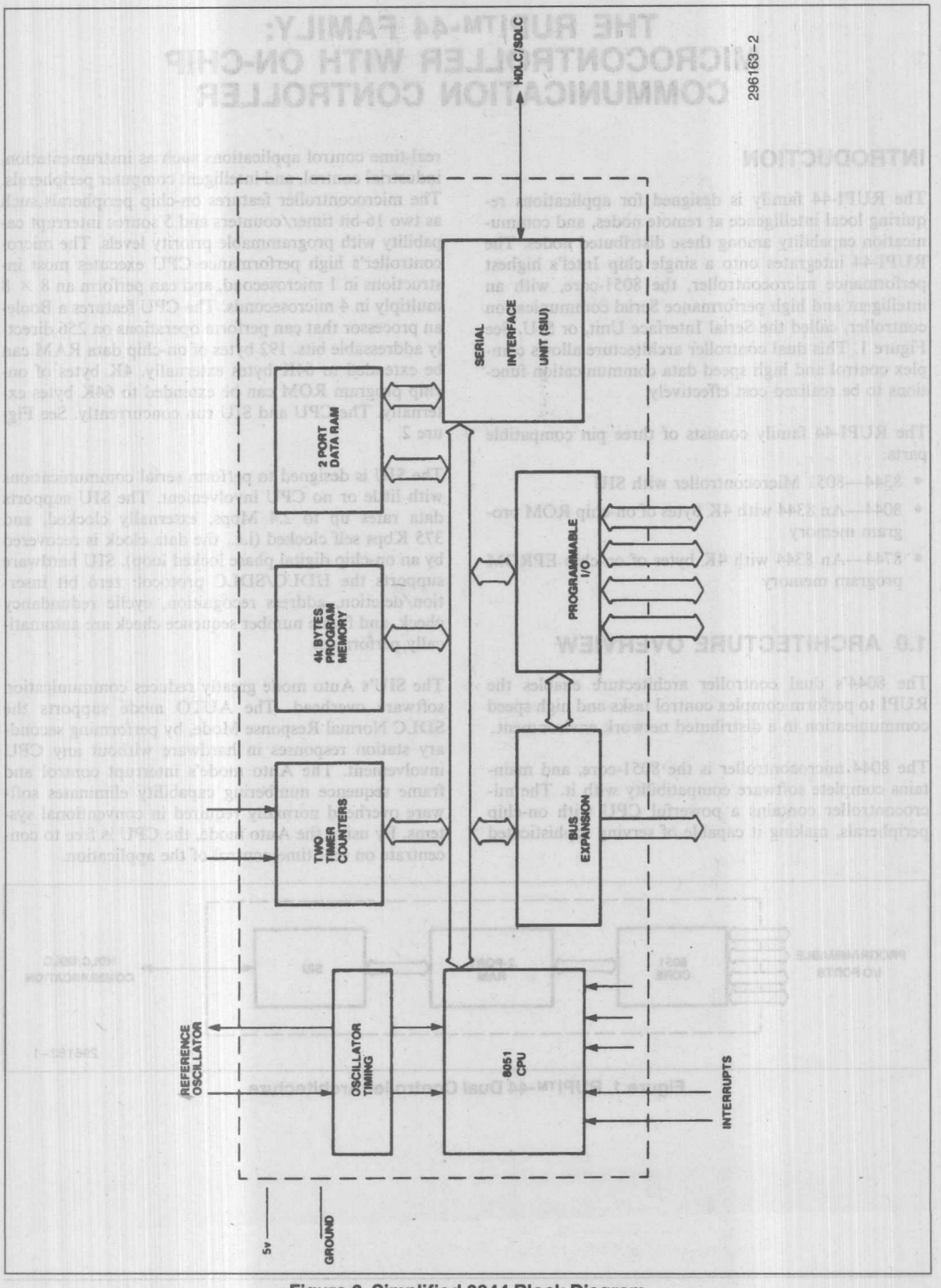


Figure 2. Simplified 8044 Block Diagram

2.0 THE HDLC/SDLC PROTOCOLS

2.1 HDLC/SDLC Advantages over Async

The High Level Data Link Control, HDLC, is a standard communication link control established by the International Standards Organization (ISO). SDLC is a subset of HDLC.

HDLC and SDLC are both well recognized standard serial protocols. The Synchronous Data Link Control, SDLC, is an IBM standard communication protocol. IBM originally developed SDLC to provide efficient, reliable and simple communication between terminals and computers.

The major advantages of SDLC/HDLC over Asynchronous communications protocol (Async):

- SIMPLE: Data Transparency
- EFFICIENT: Well Defined Message-Level Operation
- RELIABLE: Frame Check Sequence and Frame Numbering

The SDLC reduces system complexity. HDLC/SDLC are "data transparent" protocols. Data transparency means that an arbitrary data stream can be sent without concern that some of the data could be mistaken for a protocol controller. Data transparency relieves the communication controller having to detect special characters.

SDLC/HDLC provides more data throughout than Async. SDLC/HDLC runs at Message-level Operation which transmits multiple bytes within the frame, whereas Async is based on character-level operation. Async transmits or receives a character at a time. Since Async requires start and stop bits in every transmission, there is a considerable waste of overhead compared to SDLC/HDLC.

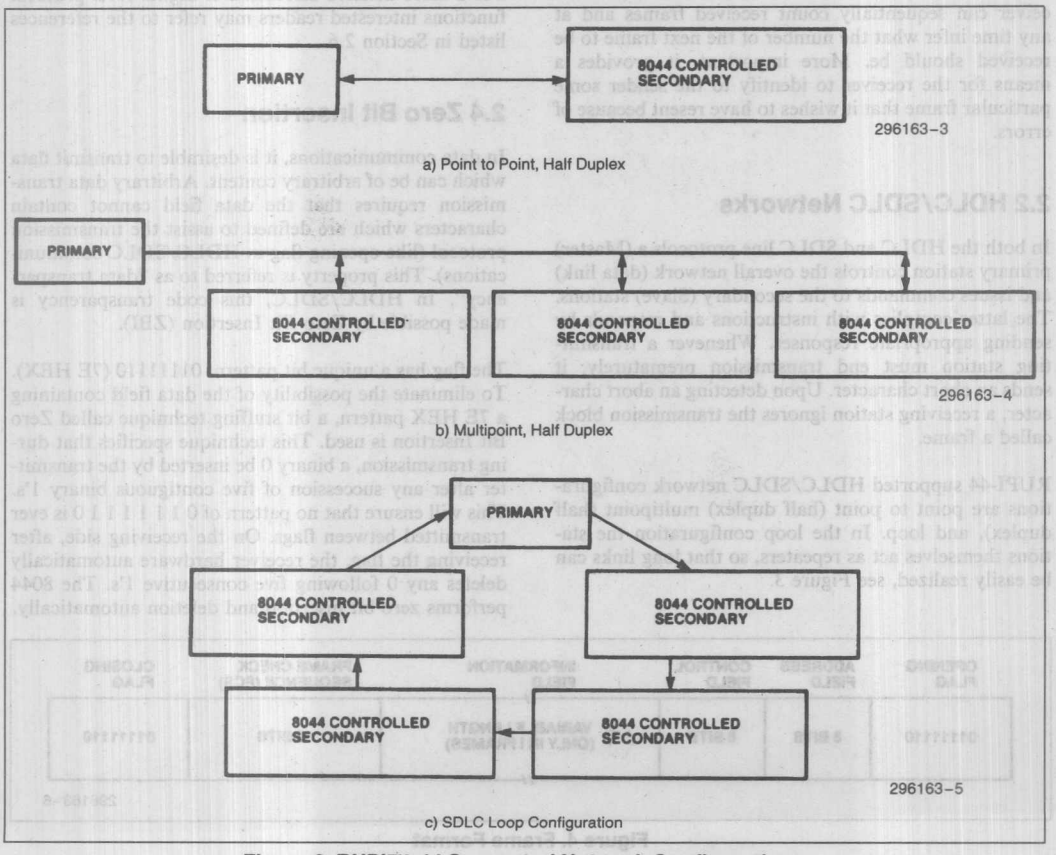


Figure 3. RUPI™-44 Supported Network Configurations

Due to SDLC/HDLC's well delineated field (see Figure 4) the CPU does not have to interpret character by character to determine control field and information field. In the case of Async, CPU must look at each character to interpret what it means. The practical advantage of such feature is straight forward use of DMA for information transfer.

In addition, SDLC/HDLC further improves Data throughput using implied Acknowledgement of transferred information. A station using SDLC/HDLC may acknowledge previously received information while transmitting different information in the same frame. In addition, up to 7 messages may be outstanding before an acknowledgement is required.

The HDLC/SDLC protocol can be used to realize reliable data links. Reliable Data transmission is ensured at the bit level by sending a frame check sequence, cyclic redudancy checking, within the frame. Reliable frame transmission is ensured by sending a frame number identification with each frame. This means that a receiver can sequentially count received frames and at any time infer what the number of the next frame to be received should be. More important, it provides a means for the receiver to identify to the sender some particular frame that it wishes to have resent because of errors.

2.2 HDLC/SDLC Networks

In both the HDLC and SDLC line protocols a (Master) primary station controls the overall network (data link) and issues commands to the secondary (Slave) stations. The latter complies with instructions and responds by sending appropriate responses. Whenever a transmitting station must end transmission prematurely, it sends an abort character. Upon detecting an abort character, a receiving station ignores the transmission block called a frame.

RUPI-44 supported HDLC/SDLC network configurations are point to point (half duplex) multipoint (half duplex), and loop. In the loop configuration the stations themselves act as repeaters, so that long links can be easily realized, see Figure 3.

2.3 Frames

An HDLC/SDLC frame consists of five basic fields: Flag, Address, Control, Data and Error Detection. A frame is bounded by flags—opening and closing flags. An address field is 8 bits wide in SDLC, extendable to 2 or more bytes in HDLC. The control field is also 8 bits wide, extendable to two bytes in HDLC. The SDLC data field or information field may be any number of bytes. The HDLC data field may or may not be on an 8 bit boundary. A powerful error detection code called Frame Check Sequence contains the calculated CRC (Cycle Redundancy Code) for all the bits between the flags. See Figure 4.

In HDLC and SDLC are three types of frames; an Information Frame is used to transfer data, a Supervisory Frame is used for control purposes, and a Nonsequenced Frame is used for initialization and control of the secondary stations.

For a more detailed discussion of higher level protocol functions interested readers may refer to the references listed in Section 2.6.

2.4 Zero Bit Insertion

In data communications, it is desirable to transmit data which can be of arbitrary content. Arbitrary data transmission requires that the data field cannot contain characters which are defined to assist the transmission protocol (like opening flag in HDLC/SDLC communications). This property is referred to as "data transparency". In HDLC/SDLC, this code transparency is made possible by Zero Bit Insertion (ZBI).

The flag has a unique bit pattern: 01111110 (7E HEX). To eliminate the possibility of the data field containing a 7E HEX pattern, a bit stuffing technique called Zero Bit Insertion is used. This technique specifies that during transmission, a binary 0 be inserted by the transmitter after any succession of five contiguous binary 1's. This will ensure that no pattern of 0 1 1 1 1 1 1 0 is ever transmitted between flags. On the receiving side, after receiving the flag, the receiver hardware automatically deletes any 0 following five consecutive 1's. The 8044 performs zero bit insertion and deletion automatically.

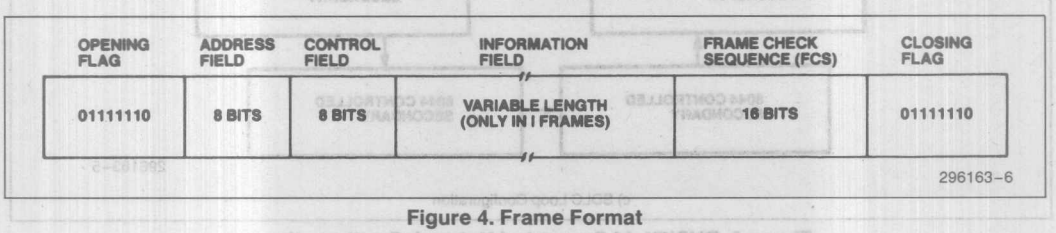


Figure 4. Frame Format

2.5 Non-return to Zero Inverted (NR21)

NRZI is a method of clock and data encoding that is well suited to the HDLC/SDLC protocol. It allows HDLC/SDLC protocols to be used with low cost asynchronous modems. NRZI coding is done at the transmitter to enable clock recovery from the data at the receiver terminal by using standard digital phase locked loop (DPLL) techniques. NRZI coding specifies that the signal condition does not change for transmitting a 1, while a 0 causes a change of state. NRZI coding ensures that an active data line will have a transition at least every 5-bit times (recall Zero Bit Insertion), while contiguous 0's will cause a change of state. Thus, ZBI and NRZI encoding makes it possible for the 8044's on-chip DPLL to recover a receive clock (from received data) synchronized to the received data and at the same time ensure data transparency.

2.6 References

1. *IBM Synchronous Data Link Control General Information GA27-3093-2 File No. GENL-09.*
2. *Standard Network Access Protocol Specification, DATAPAC Trans-Canada Telephone System CCG111.*
3. *IBM 3650 Retail Store System Loop Interface OEM Information, IBM, GA27-3098-0.*
4. *Guidebook to Data Communications, Training Manual, Hewlett-Packard 5955-1715.*
5. *"Serial Backplane Suits Multiprocessor Architectures", Mike Webb, Computer Design, July 1984, pp. 85–96.*
6. *"Serial Bus Simplifies Distributed Control", P.D. MacWilliams, Control Engineering, June 1984, pp. 101–104.*
7. *"Chips Support Two Local Area Networks", Bob Dahlberg, Computer Design, May 1984, pp. 107–114.*
8. *"Build a VLSI-based Workstation for the Ethernet Environment", Mike Webb, EDN, 23 February 1984, pp. 297–307.*
9. *"Networking With the 8044", Young Sohn & Charles Gopen, Digital Design, May 1984, pp. 136–137.*

3.0 RUPI™-44 DESIGN SUPPORT

3.1 Design Tool Support

A critical design consideration is time to market. Intel provides a sophisticated set of design tools to speed hardware and software development time of 8044 based products. These include ICE-44, ASM-51, PL/M-51, and EMV-44.



Figure 5. RUPI™-44 Development Support Configuration Intellec® System, ICE™-44 Buffer Box, and ICE-44 Module Plugged into a User Prototype Board

A primary tool is the 8044 In Circuit Emulator, called ICE-44. See Figure 5. In conjunction with Intel's Intellec® Microprocessor Development System, the ICE-44 emulator allows hardware and software development to proceed interactively. This approach is more effective than the traditional method of independent hardware and software development followed by system integration. With the ICE-44 module, prototype hardware can be added to the system as it is designed. Software and hardware integration occurs while the product is being developed.

The ICE-44 emulator assists four stages of development:

1) Software Debugging

   It can be operated without being connected to the user's system before any of the user's hardware is available. In this stage ICE-44 debugging capabilities can be used in conjunction with the Intellec text editor and 8044 macroassembler to facilitate program development.

2) Hardware Development

   The ICE-44 module's precise emulation characteristics and full-speed program RAM make it a valuable tool for debugging hardware, including the time-critical SDLC serial port, parallel port, and timer interfaces.

Integration of software and hardware can begin when any functional element of the user system hardware is connected to the 8044 socket. As each section of the user's hardware is completed, it is added to the prototype. Thus, each section of the hardware and software is system tested in real-time operation as it becomes available.

4) System Test

When the user's prototype is complete, it is tested with the final version of the user system software. The ICE-44 module is then used for real-time emulation of the 8044 to debug the system as a completed unit.

The final product verification test may be performed using the 8744 EPROM version of the 8044 microcomputer. Thus, the ICE-44 module provides the user with the ability to debug a prototype or production system at any stage in its development.

A conversion kit, ICE-44 CON, is available to upgrade an ICE-51 module to ICE-44.

Intel's ASM-51 Assembler supports the 8044 special function registers and assembly program development. PL/M-51 provides designers with a high level language for the 8044. Programming in PL/M can greatly reduce development time, and ensure quick time to market.

These tools have recently been expanded with the addition of the EMV-44CON. This conversion kit allows you to convert an EMV-51 into an EMV-44 emulation vehicle. The resultant low cost emulator is designed for use with an iPDS Personal Development System, which also supports the ASM-51 assembler and PL/M-51. See Figure 6.

Emulation support is similar to the ICE-44 with support for Software and Hardware Development, System



Figure 6. RUPI-44 iPDS Personal Development System, EMV-44 Buffer Box, and EMV-44 Module Plugged into a User Prototype Board

Integration, and System Test. The iPDS's rugged portability and ease of use also make it an ideal system for production tests and field service of your finished design. In addition, the iPDS offers EPROM programming module for the 8744, and direct communications with the 8044-based BITBUS via an optional iSBX-344 distributed control module.

3.2 8051 Workshop

Intel provides 8051 training to its customers through the 5-day 8051 workshop. Familiarity with the 8051 and 8044 is achieved through a combination of lecture and laboratory exercises.

For designers not familiar with the 8051, the workshop is an effective way to become proficient with the 8051 architecture and capabilities.

8044 Architecture

12

8044 ARCHITECTURE

GENERAL

The 8044 is based on the 8051 core. The 8044 replaces the 8051's serial port with an intelligent HDLC/SDLC controller called the Serial Interface or SIU. Thus the differences between the two result from the 8044's increased on-chip RAM (192 bytes) and additional special function registers necessary to control the SIU. Aside from the increased memory, the SIU itself, and differences in 5 pins (for the serial port), the 8044 and 8051 are compatible.

This chapter describes the differences between the 8044 and 8051. Information pertaining to the 8051 core, eg. instruction set, port operation, EPROM programming, etc. is located in the 8051 sections of this manual.

A block diagram of the 8044 is shown in Figure 1. The pinpoint is shown on the inside front cover.

1.0 MEMORY ORGANIZATION OVERVIEW

The 8044 maintains separate address spaces for Program Memory and Data Memory. The Program Memory can be up to 64K bytes long, of which the lowest 4K bytes are in the on-chip ROM.

If the $\overline{EA}$ pin is held high, the 8044 executes out of internal ROM unlwess the Program Counter exceeds 0FFFH. Fetches from locations 1000H through FFFFH are directed to external Program Memory.

If the $\overline{EA}$ pin is held low, the 8044 fetches all instructions from external Program Memory.

The Data Memory consists of 192 bytes of on-chip RAM, plus 35 Special Function Registers, in addition to which the device is capable of accessing up to 64K bytes of external data memory.

The Program Memory uses 16-bit addresses. The external Data Memory can use erither 8-bit or 16-bit addresses. The internal Data Memory uses 8-bit addresses, which provide a 256-location address space. The lower 192 addresses access the on-chip RAM. The Special Function Registers occupy various locations in the upper 128 bytes of the same address space.

The lowest 32 bytes in the internal RAM (locations 00 through 1FH) are divided into 4 banks of registers, each bank consisting of 8 bytes. Any one of these banks can be selected to be the "working registers" of the CPU, and can be accessed by a 3-bit address in the same byte as the opcode of an instruction. Thus, a large number of instructions are one-byte instructions.

The next higher 16 bytes of the internal RAM (locations 20H through 2FH) have individually addressable bits. These are provided for use as software flags or for one-bit (Boolean) processing. This bit-addressing capability is an important feature of the 8044. In addition to the 128 individually addressable bits in RAM, twelve of the Special Function Registers also have individually addressable bits.

A memory map is shown in Figure 2.

1.1 Special Function Registers

The Special Function Registers are as follows:

*	ACC	Accumulator (A Register)
*	B	B Register
*	PSW	Program Status Word
	SP	Stack Pointer
	DPTR	Data Pointer (consisting of DPH AND DPL)
*	P0	Port 0
*	P1	Port 1
*	P2	Port 2
*	P3	Port 3
*	IP	Interrupt Priority
*	IE	Interrupt Enable
	TMOD	Timer/Counter Mode
*	TCON	Timer/Counter Control
	TH0	Timer/Counter 0 (high byte)
	TL0	Timer/Counter 0 (low byte)
	TH1	Timer/Counter 1 (high byte)
	TL1	Timer/Counter 1 (low byte)
	SMD	Serial Mode
*	STS	Status/Command
*	NSNR	Send/Receive Count
	STAD	Station Address
	TBS	Transmit Buffer Start Address
	TBL	Transmit Buffer Length
	TCB	Transmit Control Byte
	RBS	Receive Buffer Start Address
	RBL	Receive Buffer Length
	RFL	Received Field Length
	RCB	Received Control Byte
	DMA CNT	DMA Count
	FIFO	FIFO (three bytes)
	SIUST	SIU State Counter
	PCON	Power Control

The registers marked with * are both byte- and bit-addressable.

Order Number: 296164-001

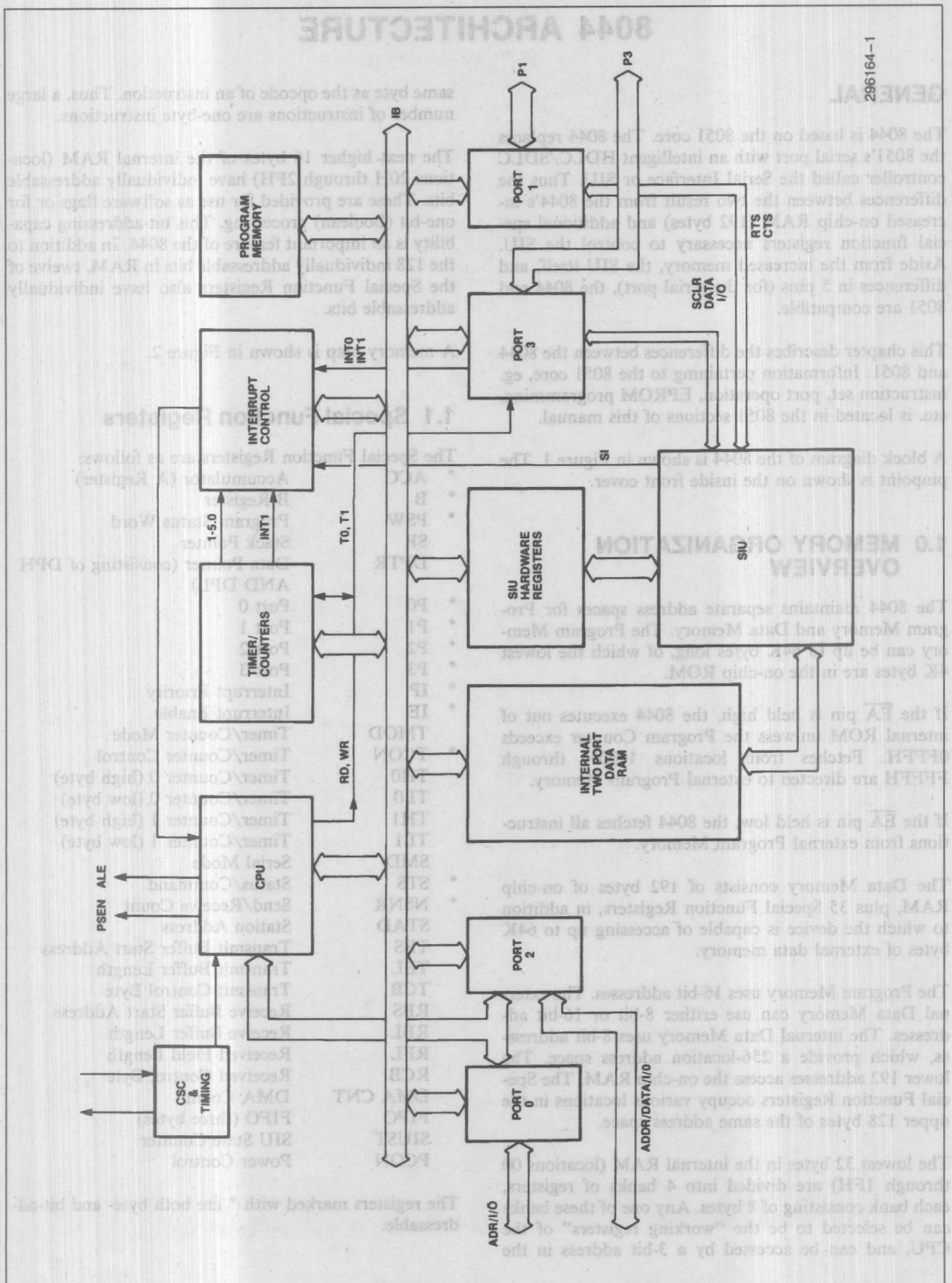


Figure 1. RUPI™ Block Diagram

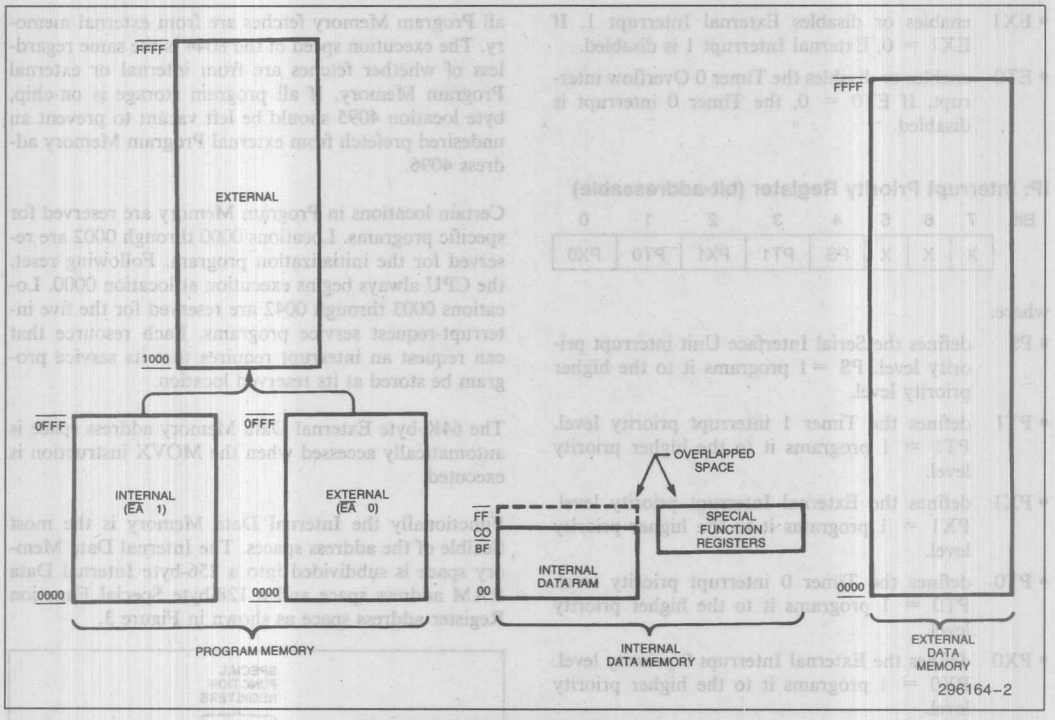


Figure 2. RUPI™-44 Memory Map

Stack Pointer

The Stack Pointer is 8 bits wide. The stack can reside anywhere in the 192 bytes of on-chip RAM. When the 8044 is reset, the stack pointer is initialized to 07H. When executing a PUSH or a CALL, the stack pointer is incremented before data is stored, so the stack would begin at location 08H.

1.2 Interrupt Control Registers

The Interrupt Request Flags are as listed below:

Source	*Request Flag*	*Location*
External Interrupt 0	$\overline{INT0}$, if IT0 = 0	P3.2
	IE0, if IT0 = 1	TCON.1
Timer 0 Overflow	TF0	TCON.5
External Interrupt 1	$\overline{INT1}$, if IT1 = 0	P3.3
	IE1, if IT1 = 1	TCON.3
Timer 1 Overflow	TF1	TCON.7
Serial Interface Unit	SI	STS.4

External Interrupt control bits IT0 and IT1 are in TCON.0 and TCON.2, respectively. Reset leaves all flags inactive, with IT0 and IT1 cleared.

All the interrupt flags can be set or cleared by software, with the same effect as by hardware.

The Enable and Priority Control Registers are shown below. All of these control bits are set or cleared by software. All are cleared by reset.

IE: Interrupt Enable Register (bit-addressable)

Bit:	7	6	5	4	3	2	1	0
	EA	X	X	ES	ET1	EX1	ET0	EX0

where:

- EA disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
- ES enables or disables the Serial Interface Unit interrupt. If ES = 0, the Serial Interface Unit interrupt is disabled.
- ET1 enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.

- EX1 enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
- ET0 enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.

IP: Interrupt Priority Register (bit-addressable)

Bit:	7	6	5	4	3	2	1	0
	X	X	X	PS	PT1	PX1	PT0	PX0

where:

- PS defines the Serial Interface Unit interrupt priority level. PS = 1 programs it to the higher priority level.
- PT1 defines the Timer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
- PX1 defines the External Interrupt priority level. PX1 = 1 programs it to the higher priority level.
- PT0 defines the Timer 0 interrupt priority level. PT0 = 1 programs it to the higher priority level.
- PX0 defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

2.0 MEMORY ORGANIZATION DETAILS

In the 8044 family the memory is organized over three address spaces and the program counter. The memory spaces shown in Figure 2 are the:

- 64K-byte Program Memory address space
- 64K-byte External Data Memory address space
- 320-byte Internal Data Memory address space

The 16-bit Program Counter register provides the 8044 with its 64K addressing capabilities. The Program Counter allows the user to execute calls and branches to any location within the Program Memory space. There are no instructions that permit program execution to move from the Program Memory space to any of the data memory spaces.

In the 8044 and 8744 the lower 4K of the 64K Program Memory address space is filled by internal ROM and EPROM, respectively. By tying the $\overline{EA}$ pin high, the processor can be forced to fetch from the internal ROM/EPROM for Program Memory addresses 0 through 4K. Bus expansion for accessing Program Memory beyond 4K is automatic since external instruction fetches occur automatically when the Program Counter increases above 4095. If the $\overline{EA}$ pin is tied low all Program Memory fetches are from external memory. The execution speed of the 8044 is the same regardless of whether fetches are from internal or external Program Memory. If all program storage is on-chip, byte location 4095 should be left vacant to prevent an undesired prefetch from external Program Memory address 4096.

Certain locations in Program Memory are reserved for specific programs. Locations 0000 through 0002 are reserved for the initialization program. Following reset, the CPU always begins execution at location 0000. Locations 0003 through 0042 are reserved for the five interrupt-request service programs. Each resource that can request an interrupt requires that its service program be stored at its reserved location.

The 64K-byte External Data Memory address space is automatically accessed when the MOVX instruction is executed.

Functionally the Internal Data Memory is the most flexible of the address spaces. The Internal Data Memory space is subdivided into a 256-byte Internal Data RAM address space and a 128-byte Special Function Register address space as shown in Figure 3.

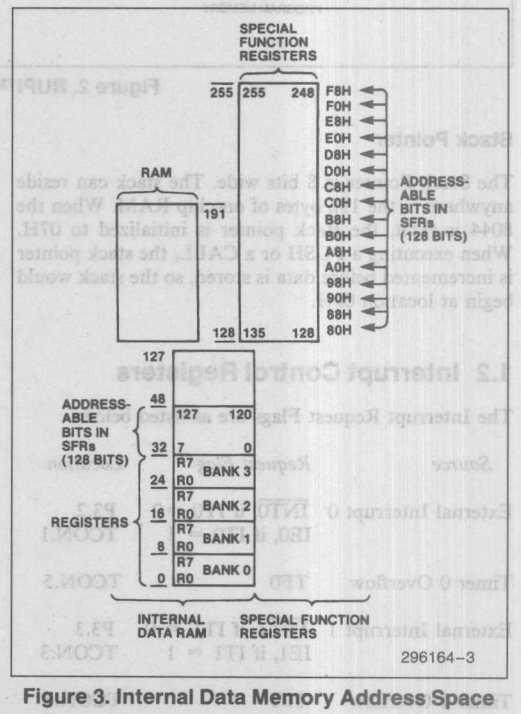


Figure 3. Internal Data Memory Address Space

The Internal Data RAM address space is 0 to 255. Four 8-Register Banks occupy locations 0 through 31. The stack can be located anywhere in the Internal Data RAM address space. In addition, 128 bit locations of the on-chip RAM are accessible through Direct Addressing. These bits reside in Internal Data RAM at byte locations 32 through 47. Currently locations 0 through 191 of the Internal Data RAM address space are filled with on-chip RAM.

The stack depth is limited only by the available Internal Data RAM, thanks to an 8-bit reloadable Stack Pointer. The stack is used for storing the Program Counter during subroutine calls and may be used for passing parameters. Any byte of Internal Data RAM or Special Function Register accessible though Direct Addressing can be pushed/popped.

The Special Function Register address space is 128 to 255. All registers except the Program Counter and the four 8-Register Banks reside here. Memory mapping the Special Function Registers allows them to be accessed as easily as internal RAM. As such, they can be operated on by most instructions. In the overlapping memory space (address 128-191), indirect addressing is used to access RAM, and direct addressing is used to access the SFR's. The SFR's at addresses 192–255 are also accessed using direct addressing. The Special Function Registers are listed in Figure 4. Their mapping in the Special Function Register address space is shown in Figures 5 and 6.

Performing a read from a location of the Internal Data memory where neither a byte of Internal Data RAM (i.e., RAM addresses 192-255) nor a Special Function Register exists will access data of indeterminable value.

Architecturally, each memory space is a linear sequence of 8-bit wide bytes. By Intel convention the storage of multi-byte address and data operands in program and data memories is the least significant byte at the low-order address and the most significant byte at the high-order address. Within byte X, the most significant bit is represented by X.7 while the least significant bit is X.0. Any deviation from these conventions will be explicitly stated in the text.

2.1 Operand Addressing

There are five methods of addressing source operands. They are Register Addressing, Direct Addressing, Register-Indirect Addressing, Immediate Addressing

ARITHMETIC REGISTERS:
Accumulator*, B register*,
Program Status Word*
POINTERS:
Stack Pointer, Data Pointer (high & low)
PARALLEL I/O PORTS:
Port 3*, Port 2*, Port 1*, Port 0*
INTERRUPT SYSTEM:
Interrupt Priority Control*,
Interrupt Enable Control*
TIMERS:
Timer Mode, Timer Control*, Timer 1 (high & low), Timer 0 (high & low)
SERIAL INTERFACE UNIT:
Transmit Buffer Start,
Transmit Buffer Length,
Transmit Control Byte,
Send Count Receive Count*,
DMA Count,
Station Address
Receive Field Length
Receive Buffer Start
Receive Buffer Length
Receive Control Byte,
Serial Mode,
Status Register.*

*Bits in these registers are bit addressable.

Figure 4. Special Function Registers

ARITHMETIC REGISTERS:
Accumulator*, B register*,
Program Status Word*
POINTERS:
Stack Pointer, Data Pointer (high & low)
PARALLEL I/O PORTS:
Port 3*, Port 2*, Port 1*, Port 0*
INTERRUPT SYSTEM:
Interrupt Priority Control*,
Interrupt Enable Control*
TIMERS:
Timer Mode, Timer Control*, Timer 1 (high & low), Timer 0 (high & low)
SERIAL INTERFACE UNIT:
Serial Mode, Status/Command*,
Send/Receive Count*, Station Address,
Transmit Buffer Start Address,
Transmit Buffer Length,
Transmit Control Byte,
Receive Buffer Start Address,
Receive Buffer Length,
Receive Field Length,
Receive Control Byte,
DMA Count,
FIFO (three bytes),
SIU Controller State Counter

*Bits in these registers are bit-addressable.

Figure 5. Mapping of Special Function Registers

and Base-Register-plus Index-Register-Indirect Addressing. The first three of these methods can also be used to address a destination operand. Since operations in the 8044 require 0 (NOP only), 1, 2, 3 or 4 operands, these five addressing methods are used in combinations to provide the 8044 with its 21 addressing modes.

Most instructions have a "destination, source" field that specifies the data type, addressing methods and operands involved. For operations other than moves, the destination operand is also a source operand. For example, in "subtract-with-borrow A, #5" the A register receives the result of the value in register A minus 5, minus C.

Most operations involve operands that are located in Internal Data Memory. The selection of the Program Memory space or External Data Memory space for a second operand is determined by the operation mnemonic unless it is an immediate operand. The subset of the Internal Data Memory being addressed is determined by the addressing method and address value. For example, the Special Function Registers can be accessed only through Direct Addressing with an address of 128-255. A summary of the operand addressing methods is shown in Figure 6. The following paragraphs describe the five addressing methods.

2.2 Register Addressing

Register Addressing permits access to the eight registers (R7-R0) of the selected Register Bank (RB). One of the four 8-Register Banks is selected by a two-bit field in the PSW. The registers may also be accessed through Direct Addressing and Register-Indirect Addressing, since the four Register Banks are mapped into the lowest 32 bytes of internal Data RAM as shown in Figures 9 and 10. Other Internal Data Memory locations that are addressed as registers are A, B, C, AB and DPTR.

2.3 Direct Addressing

Direct Addressing provides the only means of accessing the memory-mapped byte-wide Special Function Registers and memory mapped bits within the Special Function Registers and Internal Data RAM. Direct Addressing of bytes may also be used to access the lower 128 bytes of Internal Data RAM. Direct Addressing of bits gains access to a 128 bit subset of the Special Function Registers as shown in Figures 5, 6, 9, and 10.

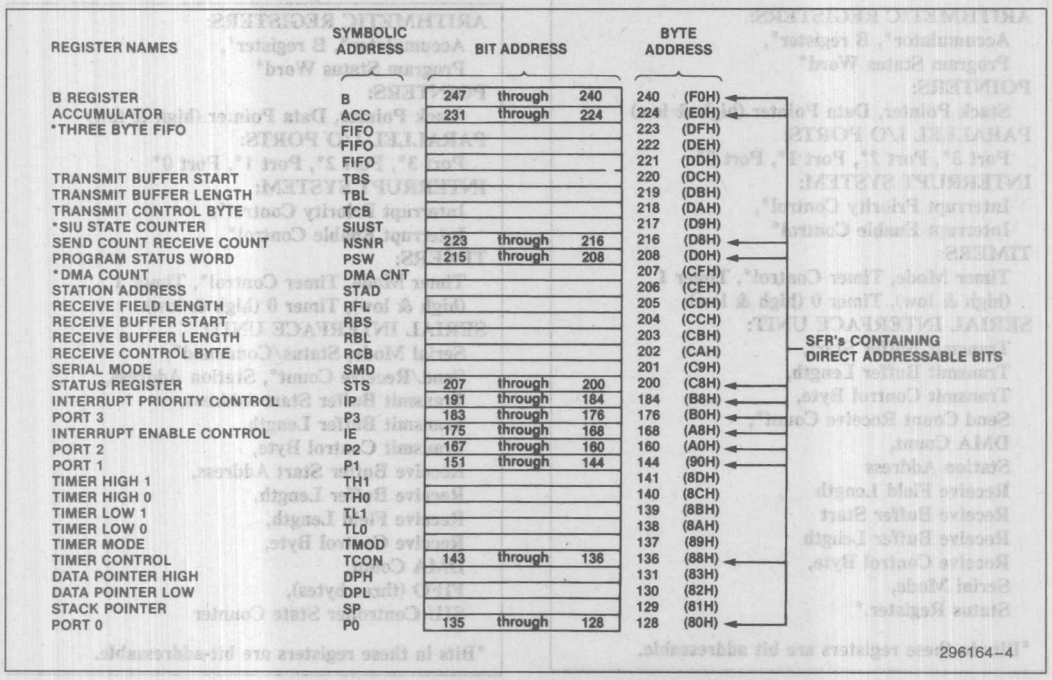


Figure 6. Mapping of Special Function Registers

Direct Byte Address	Bit Address (MSB)							(LSB)	Hardware Register Symbol
240	F7	F6	F5	F4	F3	F2	F1	F0	8
224	E7	E6	E5	E4	E3	E2	E1	E0	ACC
	NS2	NS1	NS0	SES	NR2	NR1	NR0	SER	
216	DF	DE	DD	DC	DB	DA	D9	D8	NSNR
	CY	AC	FO	RS1	RSO	OV		P	
204	D7	D6	D5	D4	D3	D2	D1	D0	PSW
	TBF	RE	RTS	SI	BV	CPB	AM	RBP	
200	CF	CE	CD	CC	CB	CA	C9	C8	STS
				PS	PT1	PX1	PT0	PX0	
184	—	—	—	BC	BB	BA	B9	B8	1P
176	B7	B6	B5	B4	B3	B2	B1	B0	P3
	EA			E5	ET1	EX1	ET0	EX0	
168	AF	—	—	AC	AB	AA	A9	A8	1E
160	A7	A6	A5	A4	A3	A2	A1	A0	P2
144	97	96	95	94	93	92	91	90	P1
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
136	8F	8E	8D	8C	8B	8A	89	88	TCON
128	87	86	85	84	83	82	81	80	P0

Figure 7. Special Function Register Bit Address

- **Register Addressing**
  - — R7-R0
  - — A, B, C (bit), AB (two bytes), DPTR (double byte)
- **Direct Addressing**
  - — Lower 128 bytes of Internal Data RAM
  - — Special Function Registers
  - — 128 bits in subset of Special Function Register address space
- **Register-Indirect Addressing**
  - — Internal Data RAM [@R1, @R0, @SP (PUSH and POP only)]
  - — Least Significant Nibbles in Internal Data RAM (@R1, @R0)
  - — External Data Memory (@R1, @R0, @DPTR)
- **Immediate Addressing**
  - — Program Memory (in-code constant)
- **Base-Register-plus Index-Register-Indirect Addressing**
  - — Program Memory (@ DPTR + A, @ PC + A)

Figure 8. Operand Addressing Methods

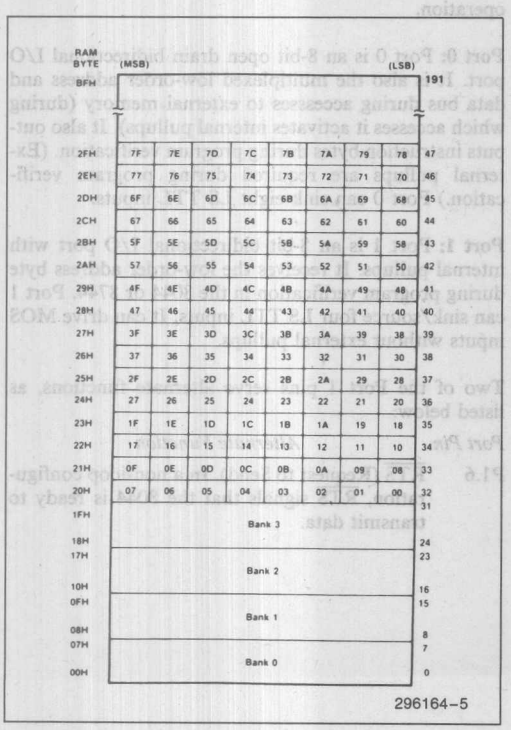


Figure 9. RAM Bit Address

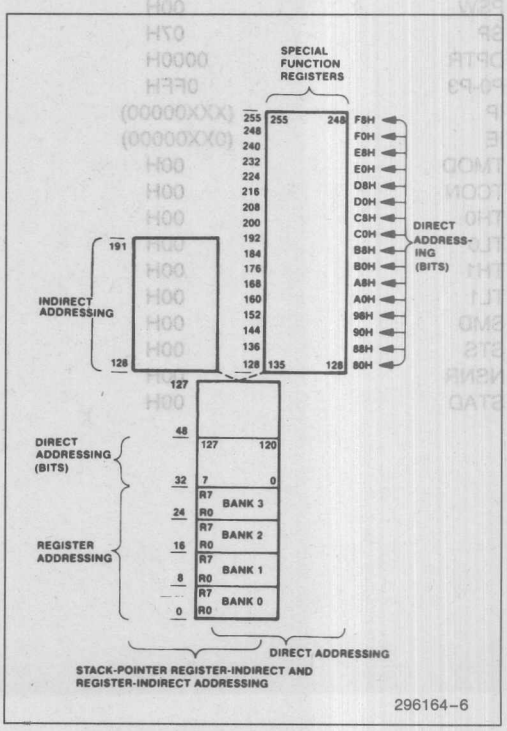


Figure 10. Addressing Operands in Internal Data Memory

Register-Indirect Addressing using the content of R1 or R0 in the selected Register Bank, or using the content of the Stack Pointer (PUSH and POP only), addresses the Internal Data RAM. Register-Indirect Addressing is also used for accessing the External Data Memory. In this case, either R1 or R0 in the selected Register Bank may be used for accessing locations within a 256-byte block. The block number can be preselected by the contents of a port. The 16-bit Data Pointer may be used for accessing any location within the full 64K external address space.

3.0 RESET

Reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods) *while the oscillator is running.* The CPU responds by executing an internal reset. It also configures the ALE and PSEN pins as inputs. (They are quasi-bidirectional.) The internal reset is executed during the second cycle in which RST is high and is repeated every cycle until RST goes low. It leaves the internal registers as follows:

Register	Content
PC	0000H
A	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	0FFH
IP	(XXX00000)
IE	(0XX00000)
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SMD	00H
STS	00H
NSNR	00H
STAD	00H
TBS	00H
TBL	00H
TCB	00H
RBS	00H
RBL	00H
RFL	00H
RCB	00H
DMA CNT	00H
FIFO1	00H
FIFO2	00H
FIFO3	00H
SIUST	01H
PCON	(0XXXXXXX)

The internal RAM is not affected by reset. When VCC is turned on, the RAM content is indeterminate unless VPD was applied prior to VCC being turned off (see Power Down Operation.)

4.0 RUPI™-44 FAMILY PIN DESCRIPTION

VSS: Circuit ground potential.

VCC: Supply voltage during programming (of the 8744), verification (of the 8044 or 8744), and normal operation.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. It is also the multiplexed low-order address and data bus during accessses to external memory (during which accesses it activates internal pullups). It also outputs instruction bytes during program verification. (External pullups are required during program verification.) Port 0 can sink eight LS TTL inputs.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. It receives the low-order address byte during program verification in the 8044 or 8744. Port 1 can sink/source four LS TTL inputs, It can drive MOS inputs without external pullups.

Two of the Port 1 pins serve alternate functions, as listed below:

Port Pin	*Alternate Function*
P1.6	$\overline{RTS}$ (Request to Send). In a non-loop configuration, $\overline{RTS}$ signals that the 8044 is ready to transmit data.

8044 Serial Interface

13

THE RUPI™-44 SERIAL INTERFACE UNIT

SERIAL INTERFACE

The serial interface provides a high-performance communication link. The protocol used for this communication is based on the IBM Synchronous Data Link Control (SDLC). The serial interface also supports a subset of the ISO HDLC (International Standards Organization High-Level Data Link Control) protocol.

The SDLC/HDLC protocols have been accepted as standard protocols for many high-level teleprocessing systems. The serial interface performs many of the functions required to service the data link without intervention from the 8044's own CPU. The programmer is free to concentrate on the 8044's function as a peripheral controller, rather than having to deal with the details of the communication process.

Five pins on the 8044 are involved with the serial interface:

Pin 7	$\overline{RTS}$/P16
Pin 8	$\overline{CTS}$/P17
Pin 10	I/$\overline{O}$/RXD/P30
Pin 11	DATA/TXD/P31
Pin 15	SCLK/T1//P35

Figure 1 is a functional block diagram of the serial interface unit (SIU). More details on the SIU hardware are given later in this chapter.

1.0 DATA LINK CONFIGURATIONS

The serial interface is capable of operating in three serial data link configurations:

1) Half-Duplex, point-to-point
2) Half-Duplex, multipoint (with a half-duplex or full-duplex primary)
3) Loop

Figure 2 shows these three configurations. The RTS (Request to Send) and CTS (Clear to Send) hand-shaking signals are available in the point-to-point and multipoint configurations.

2.0 DATA CLOCKING OPTIONS

The serial interface can operate in an externally clocked mode or in a self clocked mode.

Externally Clocked Mode

In the externally clocked mode, a common Serial Data Clock (SCLK on pin 15) synchronizes the serial bit stream. This clock signal may come from the master CPU or primary station, or from an external phase-locked loop local to the 8044. Figure 3 illustrates the timing relationships for the serial interface signals when the externally clocked mode is used in point-to-point and multipoint data link configurations.

Incoming data is sampled at the rising edge of SCLK, and outgoing data is shifted out at the falling edge of SCLK. More detailed timing information is given in the 8044 data sheet.

Self Clocked (Asynchronous) Mode

The self clocked mode allows data transfer without a common system data clock. Using an on-chip DPLL (digital phase locked loop) the serial interface recovers the data clock from the data stream itself. The DPLL requires a reference clock equal to either 16 times or 32 times the data rate. This reference clock may be externally supplied or internally generated. When the serial interface generates this clock internally, it uses either the 8044's internal logic clock (half the crystal frequency's PH2) or the "timer 1" overflow. Figure 4 shows the serial interface signal timing relationships for the loop configuration, when the unclocked mode is used.

The DPLL monitors the received data in order to derive a data clock that is centered on the received bits. Centering is achieved by detecting all transitions of the received data, and then adjusting the clock transition (in increments of $1/16$ bit period) toward the center of the received bit. The DPLL converges to the nominal bit center within eight bit transitions, worst case.

To aid in the phase locked loop capture process, the 8044 has a NRZI (non-return-to-zero inverted) data encoding and decoding option. NRZI coding specifies that a signal does not change state for a transmitted binary 1, but does change state for a binary 0. Using the NRZI coding with zero-bit insertion, it can be guaranteed that an active signal line undergoes a transition at least every six bit times.

3.0 DATA RATES

The maximum data rate in the externally clocked mode is 2.4M bits per second (bps) a half-duplex configuration, and 1.0M in a loop configuration.

Order Number: 296165-001

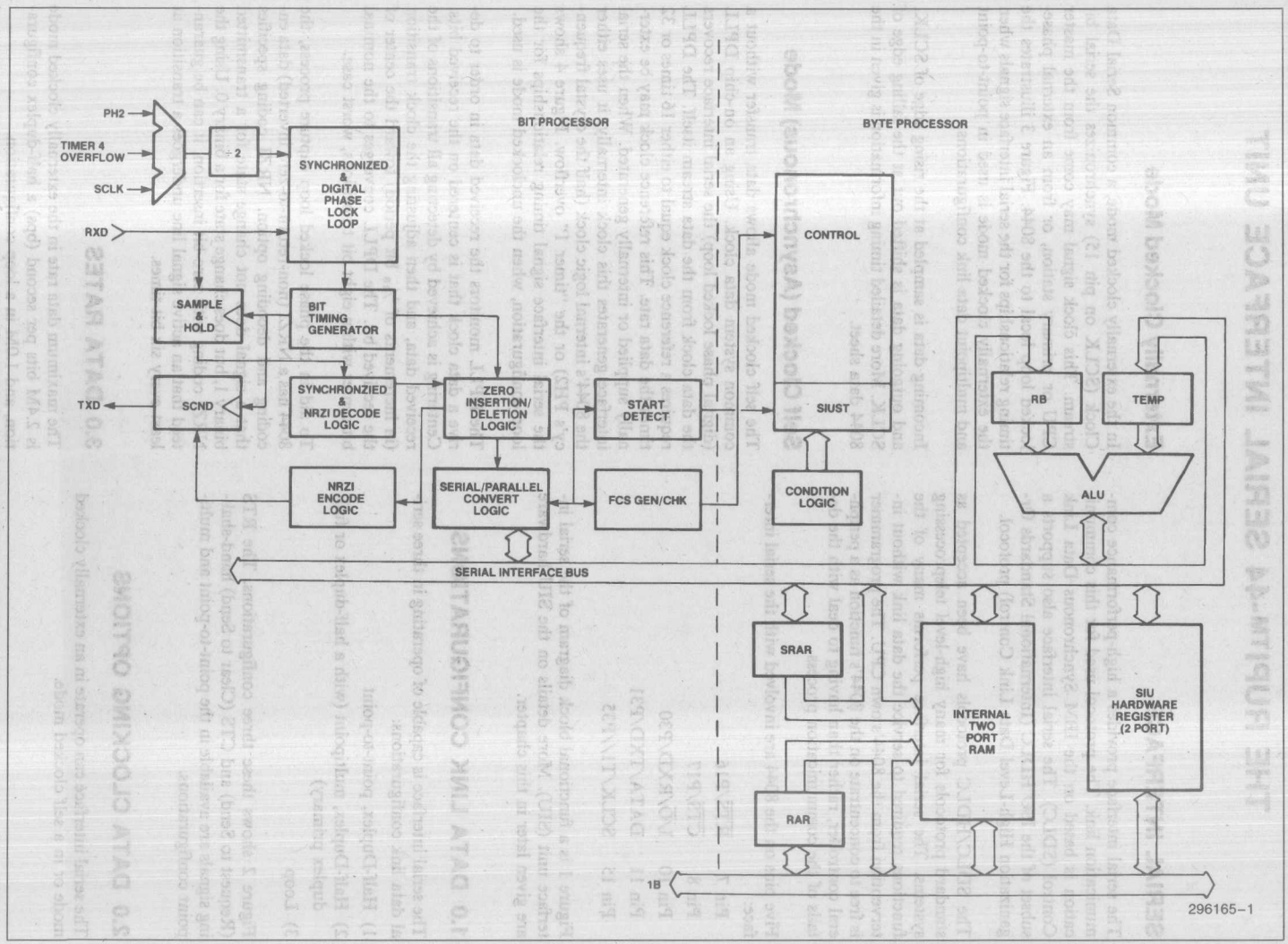


Figure 1. SIU Block Diagram

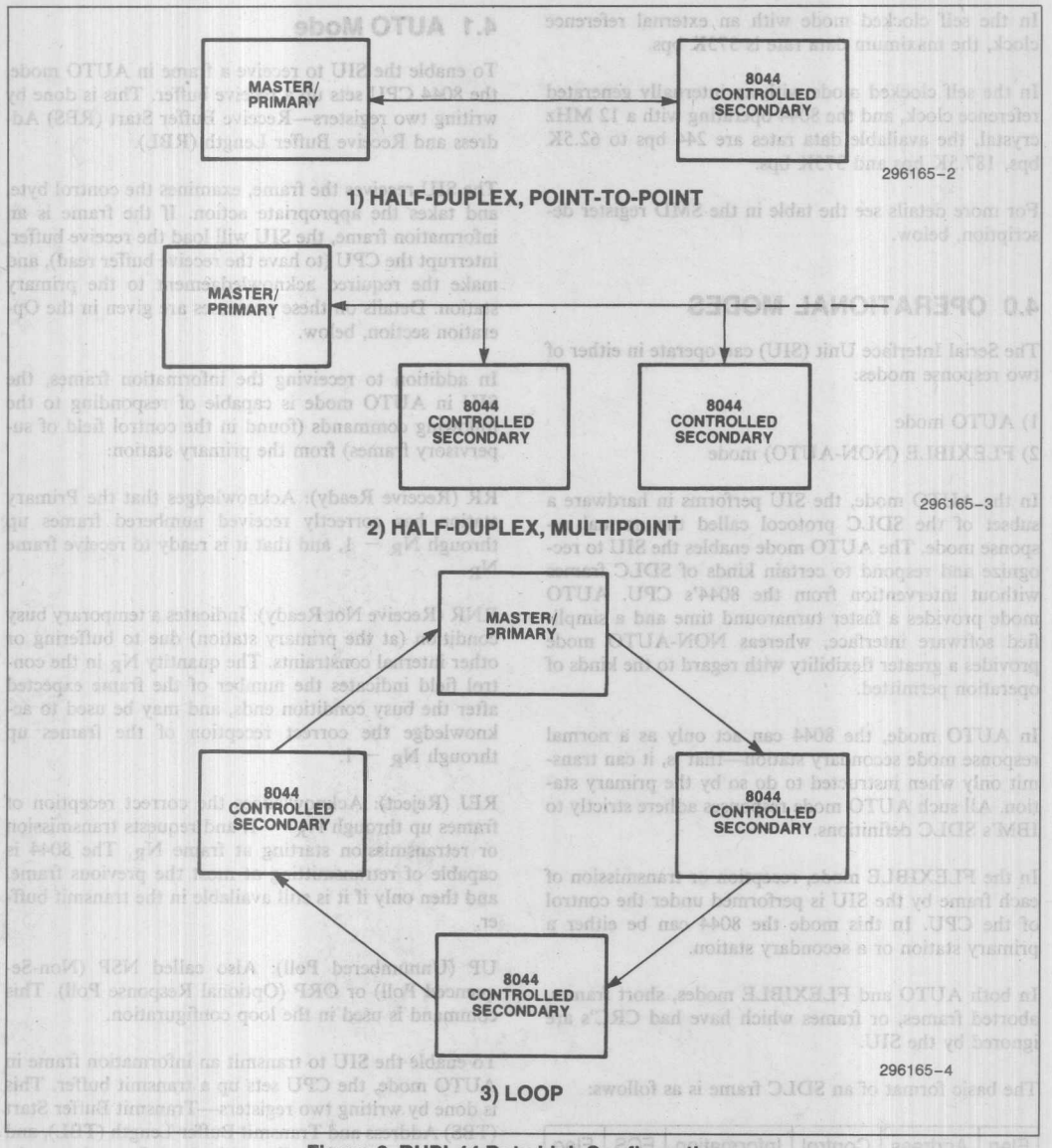


Figure 2. RUPI-44 Data Link Configurations

In the self clocked mode with an external reference clock, the maximum data rate is 375K bps.

In the self clocked mode with an internally generated reference clock, and the 8044 operating with a 12 MHz crystal, the available data rates are 244 bps to 62.5K bps, 187.5K bps and 375K bps.

For more details see the table in the SMD register description, below.

4.0 OPERATIONAL MODES

The Serial Interface Unit (SIU) can operate in either of two response modes:

1) AUTO mode
2) FLEXIBLE (NON-AUTO) mode

In the AUTO mode, the SIU performs in hardware a subset of the SDLC protocol called the normal response mode. The AUTO mode enables the SIU to recognize and respond to certain kinds of SDLC frames without intervention from the 8044's CPU. AUTO mode provides a faster turnaround time and a simplified software interface, whereas NON-AUTO mode provides a greater flexibility with regard to the kinds of operation permitted.

In AUTO mode, the 8044 can act only as a normal response mode secondary station—that is, it can transmit only when instructed to do so by the primary station. All such AUTO mode responses adhere strictly to IBM's SDLC definitions.

In the FLEXIBLE mode, reception or transmission of each frame by the SIU is performed under the control of the CPU. In this mode the 8044 can be either a primary station or a secondary station.

In both AUTO and FLEXIBLE modes, short frames, aborted frames, or frames which have had CRC's are ignored by the SIU.

The basic format of an SDLC frame is as follows:

Flag	Address	Control	Information	FCS	Flag

Format variations consist of omitting one or more of the fields in the SDLC frame. For example, a supervisory frame is formed by omitting the information field. Supervisory frames are used to confirm received frames, indicate ready or busy conditions, and to report errors. More details on frame formats are given in the SDLC Frame Format Options section, below.

4.1 AUTO Mode

To enable the SIU to receive a frame in AUTO mode, the 8044 CPU sets up a receive buffer. This is done by writing two registers—Receive Buffer Start (RBS) Address and Receive Buffer Length (RBL).

The SIU receives the frame, examines the control byte, and takes the appropriate action. If the frame is an information frame, the SIU will load the receive buffer, interrupt the CPU (to have the receive buffer read), and make the required acknowledgement to the primary station. Details on these processes are given in the Operation section, below.

In addition to receiving the information frames, the SIU in AUTO mode is capable of responding to the following commands (found in the control field of supervisory frames) from the primary station:

RR (Receive Ready): Acknowledges that the Primary station has correctly received numbered frames up through $N_R - 1$, and that it is ready to receive frame N_R.

RNR (Receive Not Ready): Indicates a temporary busy condition (at the primary station) due to buffering or other internal constraints. The quantity N_R in the control field indicates the number of the frame expected after the busy condition ends, and may be used to acknowledge the correct reception of the frames up through $N_R - 1$.

REJ (Reject): Acknowledges the correct reception of frames up through $N_R - 1$, and requests transmission or retransmission starting at frame N_R. The 8044 is capable of retransmitting at most the previous frame, and then only if it is still available in the transmit buffer.

UP (Unnumbered Poll): Also called NSP (Non-Sequenced Poll) or ORP (Optional Response Poll). This command is used in the loop configuration.

To enable the SIU to transmit an information frame in AUTO mode, the CPU sets up a transmit buffer. This is done by writing two registers—Transmit Buffer Start (TBS) Address and Transmit Buffer Length (TBL), and filling the transmit buffer with the information to be transmitted.

When the transmit buffer is full, the SIU can automatically (without CPU intervention) send an information frame (I-frame) with the appropriate sequence numbers, when the data link becomes available (when the 8044 is polled for information). After the SIU has transmitted the I-frame, it waits for acknowledgement from the receiving station. If the acknowledgement is

negative, the SIU retransmits the frame. If the acknowledgement is positive, the SIU interrupts the CPU, to indicate that the transmit buffer may be reloaded with new information.

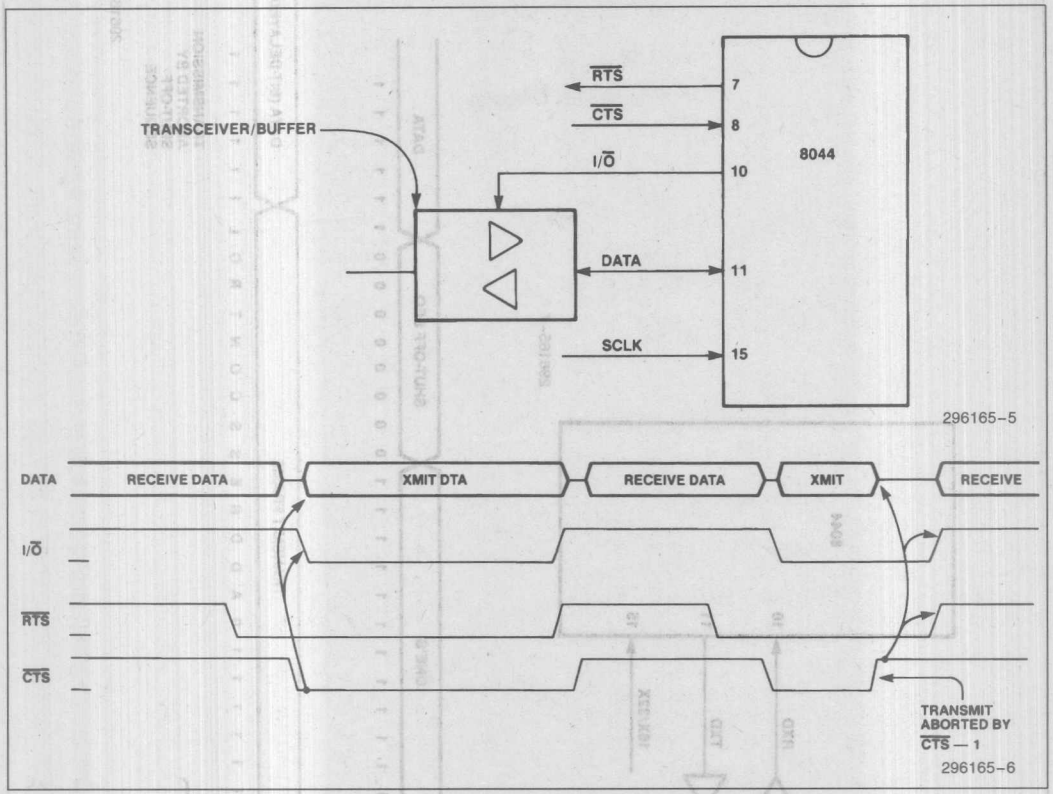


Figure 3. Serial Interface Timing—Clocked Mode

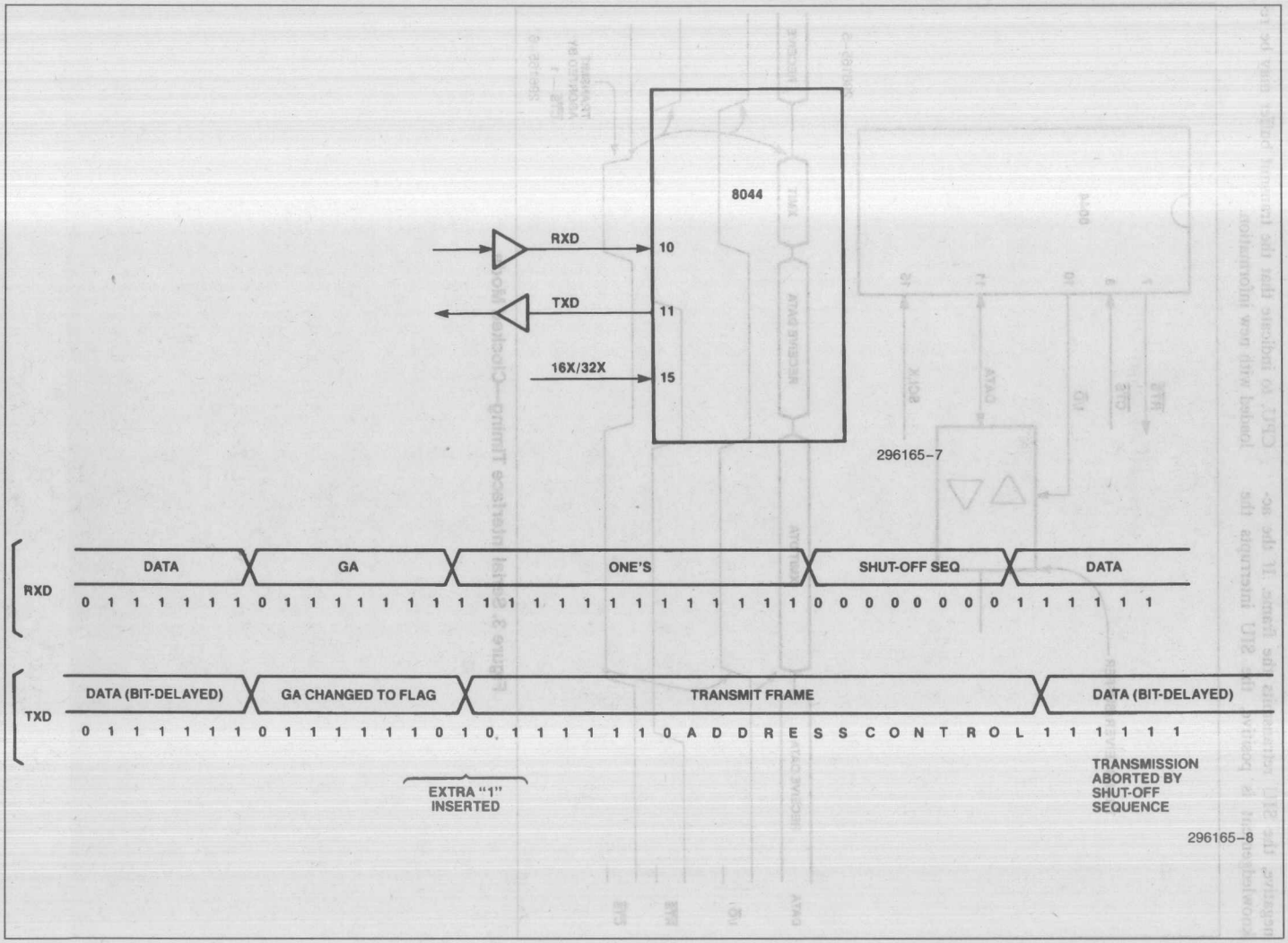


Figure 4. Serial Interface Timing—Self Clocked Mode

In addition to transmitting the information frames, the SIU in AUTO mode is capable of sending the following responses to the primary station:

RR (Receive Ready): Acknowledges that the 8044 has correctly received numbered frames up through $N_R - 1$, and that it is ready to receive frame N_R.

RNR (Receive Not Ready): Indicates a temporary busy condition (at the 8044) due to buffering or other internal constraints. The quantity N_R in the control field indicates the number of the frame expected after the busy condition ends, and acknowledges the correct reception of the frames up through $N_R - 1$.

4.2 FLEXIBLE Mode

In the FLEXIBLE (or non-auto) mode, all reception and transmission is under the control of the CPU. The full SDLC and HDLC protocols can be implemented, as well as any bit-synchronous variants of these protocols.

FLEXIBLE mode provides more flexibility than AUTO mode, but it requires more CPU overhead, and much longer recognition and response times. This is especially true when the CPU is servicing an interrupt that has higher priority than the interrupts from the SIU.

In FLEXIBLE mode, when the SIU receives a frame, it interrupts the CPU. The CPU then reads the control byte from the Receive Control Byte (RCB) register. If the received frame is an information frame, the CPU also reads the information from the receive buffer, according to the values in the Receive Buffer Start (RBS) address register and the Received Field Length (RFL) register.

In FLEXIBLE mode, the 8044 can initiate transmissions without being polled, and thus it can act as the primary station. To initiate transmission or to generate a response, the CPU sets up and enables the SIU. The SIU then formats and transmits the desired frame. Upon completion of the transmission, without waiting for a positive acknowledgement from the receiving station, the SIU interrupts the CPU.

5.0 8044 FRAME FORMAT OPTIONS

As mentioned above, variations on the basic SDLC frame consist of omitting one or more of the fields. The choice of which fields to omit, as well as the selection of AUTO mode versus FLEXIBLE mode, is specified by the settings of the following three bits in the Serial Mode Register (SMD) and the Status/Control Register (STS):

SMD Bit 0: NFCS (No Frame Check Sequence)

SMD Bit 1: NB (Non-Buffered Mode—No Control Field)

STS Bit 1: AM (AUTO Mode or Addressed Mode)

Figure 5 shows how these three bits control the frame format.

The following paragraphs discuss some properties of the standard SDLC format, and the significance of omitting some of the fields.

5.1 Standard SDLC Format

The standard SDLC format consists of an opening flag, an 8-bit address field, an 8-bit control field, an n-byte information field, a 16-bit Frame Check Sequence (FCS), and a closing flag. The FCS is based on the CCITT-CRC polynominal ($X^{16} + X^{12} + X^5 + 1$). The address and control fields may not be extended. Within the 8044, the address field is held in the Station Address (STAD) register, and the control field is held in the Receive Control Byte (RCB) or Transmit Control Byte (TCB) register. The standard SDLC format may be used in either AUTO mode or FLEXIBLE mode.

5.2 No Control Field (Non-Buffered Mode)

When the control field is not present, the RCB and TCB registers are not used. The information field begins immediately after the address field, or, if the address field is also absent, immediately after the opening flag. The entire information field is stored in the 8044's on-chip RAM. If there is no control field, FLEXIBLE mode must be used. Control information may, of course, be present in the information field, and in this manner the No Control Field option may be used for implementing extended control fields.

5.3 No Control Field and No Address Field

The No Address Field option is available only in conjuction with the No Control Field option. The STAD, RCB, and TCB registers are not used. When both these fields are absent, the information field begins immediately after the opening flag. The entire information field is stored in on-chip RAM. FLEXIBLE mode must be used. Formats without an address field have the following applications:

Point-to-point data links (where no addressing is necessary)

Monitoring line activity (receiving all messages regardless of the address field)

Extended addressing

FRAME OPTION	NFCS	NB	AM	FRAME FORMAT
Standard SDLC FLEXIBLE Mode	0	0	0	F A C I FCS F
Standard SDLC AUTO Mode	0	0	1	F A C I FCS F
No Control Field FLEXIBLE Mode	0	1	1	F A I FCS F
No Control Field No Address Field FLEXIBLE Mode	0	1	0	F I FCS F
No FCS Field FLEXIBLE Mode	1	0	0	F A C I F
No FCS Field AUTO Mode	1	0	1	F A C I F
No FCS Field No Control Field FLEXIBLE Mode	1	1	1	F A I F
No FCS Field No Control Field No Address Field FLEXIBLE Mode	1	1	0	F I F

Key to Abbreviations:

F = Flag (01111110)
A = Address Field
C = Control Field
I = Information Field
FCS = Frame Check Sequence

NOTE:
The AM bit is AUTO mode control bit when NB = 0, and Address Mode control bit when NB = 1.

Figure 5. Frame Format Options

5.4 No FCS Field

In the normal case (NFCS = 0), the last 16 bits before the closing flag are the Frame Check Sequence (FCS) field. These bits are not stored in the 8044's RAM. Rather, they are used to compute a cyclic redundancy check (CRC) on the data in the rest of the frame. A received frame with a CRC error (incorrect FCS) is ignored. In transmission, the FCS field is automatically computed by the SIU, and placed in the transmitted frame just prior to the closing flag.

The NFCS bit (SMD Bit 0) gives the user the capability of overriding this automatic feature. When this bit is set (NFCS = 1), all bits from the beginning of the information field to the beginning of the closing flag are treated as part of the information field, and are stored in the on-chip RAM. No FCS checking is done on the received frames, and no FCS is generated for the transmitted frames. The No FCS Field option may be used in conjunction with any of the other options. It is typically used in FLEXIBLE mode, although it does not strictly include AUTO mode. Use of the No FCS Field option AUTO Mode may, however, result in SDLC protocol violations, since the data integrity is not checked by the SIU.

Formats without an FCS field have the following applications:

Receiving and transmitting frames without verifying data integrity.

Using an alternate data verification algorithm.

Using an alternate CRC-16 polynomial (such as $X^{16} + X^{15} + X^2 + 1$), or a 32-bit CRC

Performing data link diagnosis by forcing false CRCs to test error detection mechanisms

In addition to the applications mentioned above, all of the format variations are useful in the support of non-standard bit-synchronous protocols.

6.0 HDLC

In addition to its support of SDLC communications, the 8044 also supports some of the capabilities of HDLC. The following remarks indicate the principal differences between SDLC and HDLC.

HDLC permits any number of bits in the information field, whereas SDLC requires a byte structure (multiple of 8 bits). The 8044 itself operates on byte boundaries, and thus it restricts fields to multiples of 8 bits.

HDLC provides functional extensions to SDLC: an unlimited address field is allowed, and extended frame number sequencing.

HDLC does not support operation in loop configurations.

7.0 SIU SPECIAL FUNCTION REGISTERS

The 8044 CPU communicates with and controls the SIU through hardware registers. These registers are accessed using direct addressing. The SIU special function registers (SIU SFRs) are of three types:

Control and Status Registers

Parameter Registers

ICE Support Registers

7.1 Control and Status Registers

There are three SIU Control and Status Registers:

Serial Mode Register (SMD)

Status/Command Register (STS)

Send/Receive Count Register (NSNR)

The SMD, STS, and NSNR registers are all cleared by system reset. This assures that the SIU will power up in an idle state (neither receiving nor transmitting).

These registers and their bit assignments are described below (see also the More Details on Registers section).

SMD: SERIAL MODE REGISTER (BYTE-ADDRESSABLE)

Bit:	7	6	5	4	3	2	1	0
	SCM2	SCM1	SCM0	NRZI	LOOP	PFS	NB	NFCS

The Serial Mode Register (Address C9H) selects the operational modes of the SIU. The 8044 CPU can both read and write SMD. The SIU can read SMD but cannot write to it. To prevent conflict between CPU and SIU access to SMD, the CPU should write SMD only when the Request To Send (RTS) and Receive Buffer Empty (RBE) bits (in the STS register) are both false (0). Normally, SMD is accessed only during initialization.

The individual bits of the Serial Mode Register are as follows:

Bit #	Name	Description
SMD.0	NFCS	No FCS field in the SDLC frame.
SMD.1	NB	Noon-Buffered mode. No control field in the SDLC frame.
SMD.2	PFS	Pre-Frame Sync mode. In this mode, the 8044 transmits two bytes before the first flag of a frame, for DPLL synchronization. If NRZI is enabled, 00H is sent; otherwise, 55H is sent. In either case, 16 pre-frame transitions are guaranteed.
SMD.3	LOOP	Loop configuration.
SMD.4	NRZI	NRZI coding option.
SMD.5	SCM0	Select Clock Mode—Bit 0
SMD.6	SCM1	Select Clock Mode—Bit 1
SMD.7	SCM2	Select Clock Mode—Bit 2

The SCM bits decode as follows:

SCM 2 1 0	Clock Mode	Data Rate (Bits/sec)*
0 0 0	Externally clocked	0–2.4M**
0 0 1	Undefined	
0 1 0	Self clocked, timer overflow	244–62.5K
0 1 1	Undefined	
1 0 0	Self clocked, external 16x	0–375K
1 0 1	Self clocked, external 32x	0–187.5K
1 1 0	Self clocked, internal fixed	375K
1 1 1	Self clocked, internal fixed	187.5K

*Based on a 12 MHz crystal frequency
**0–1M bps in loop configuration

STS: STATUS/COMMAND REGISTER (BIT-ADDRESSABLE)

Bit:	7	6	5	4	3	2	1	0
	TBF	RBE	RTS	SI	BOV	OPB	AM	RBP

The Status/Command Register (Address C8H) provides operational control of the SIU by the 8044 CPU, and enables the SIU to post status information for the CPU's access. The SIU can read STS, and can alter certain bits, as indicated below. The CPU can both read and write STS asynchronously. However, 2-cycle instructions that access STS during both cycles ('JBC/B, REL' and 'MOV /B,C.') should not be used, since the SIU may write to STS between the two CPU accesses.

The individual bits of the Status/Command Register are as follows:

Bit#	Name	Description
STS.0	RBP	Receive Buffer Protect. Inhibits writing of data into the receive buffer. In AUTO mode, RBP forces an RNR response instead of an RR.
STS.1	AM	AUTO Mode/Addressed Mode. Selects AUTO mode where AUTO mode is allowed. If NB is true, (= 1), the AM bit selects the addressed mode. AM may be cleared by the SIU.
STS.2	OPB	Optional Poll Bit. Determines whether the SIU will generate an AUTO response to an optional poll (UP with P = 0). OPB may be set or cleared by the SIU.
STS.3	BOV	Receive Buffer Overrun. BOV may be set or cleared by the SIU.
STS.4	SI	SIU Interrupt. This is one of the five interrupt sources to the CPU. The vector location = 23H. SI may be set by the SIU. It should be cleared by the CPU before returning from an interrupt routine.
STS.5	RTS	Request To Send. Indicates that the 8044 is ready to transmit or is transmitting. RTS may be read or written by the CPU. RTS may be read by the SIU, and in AUTO mode may be written by the SIU.
STS.6	RBE	Receive Buffer Empty. RBE can be thought of as Receive Enable. RBE is set to one by the CPU when it is ready to receive a frame, or has just read the buffer, and to zero by the SIU when a frame has been received.
STS.7	TBF	Transmit Buffer Full. Written by the CPU to indicate that it has filled the transmit buffer. TBF may be cleared by the SIU.

NSNR: SEND/RECEIVE COUNT REGISTER (BIT-ADDRESSABLE)

Bit:	7	6	5	4	3	2	1	0
	NS2	NS1	NS0	SES	NR2	NR1	NR0	SER

The Send/Receive Count Register (Address D8H) contains the transmit and receive sequence numbers, plus tally error indications. The SIU can both read and write NSNR. The 8044 CPU can both read and write NSNR asynchronously. However, 2-cycle instructions that access NSNR during both cycles ('JBC /B, REL', and 'MOV /B,C') should not be used, since the SIU may write to NSNR between the two 8044 CPU accesses.

The individual bits of the Send/Receive Count Register are as follows:

Bit#	Name	Description
NSNR.0	SER	Receive Sequence Error: NS (P) ≠ NR (S)
NSNR.1	NR0	Receive Sequence Counter—Bit 0
NSNR.2	NR1	Receive Sequence Counter—Bit 1
NSNR.3	NR2	Receive Sequence Counter—Bit 2
NSNR.4	SES	Send Sequence Error: NR (P) ≠ NS (S) and NR (P) ≠ NS (S) + 1
NSNR.5	NS0	Send Sequence Counter—Bit 0
NSNR.6	NS1	Send Sequence Counter—Bit 1
NSNR.7	NS2	Send Sequence Counter—Bit 2

7.2 Parameter Registers

There are eight parameter registers that are used in connection with SIU operation. All eight registers may be read or written by the 8044 CPU. RFL and RCB are normally loaded by the SIU.

The eight parameter registers are as follows:

STAD: STATION ADDRESS REGISTER (BYTE-ADDRESSABLE)

The Station Address register (Address CEH) contains the station address. To prevent access conflict, the CPU

should access STAD only when the SIU is idle (RTS = 0 and RBE = 0). Normally, STAD is accessed only during initialization.

TBS: TRANSMIT BUFFER START ADDRESS REGISTER (BYTE-ADDRESSABLE)

The Transmit Buffer Start address register (Address DCH) points to the location in on-chip RAM for the beginning of the I-field of the frame to be transmitted. The CPU should access TBS only when the SIU is not transmitting a frame (when TBF = 0).

TBL: TRANSMIT BUFFER LENGTH REGISTER (BYTE-ADDRESSABLE)

The Transmit Buffer Length register (Address DBH) contains the length (in bytes) of the I-field to be transmitted. A blank I-field (TBL = 0) is valid. The CPU should access TBL only when the SIU is not transmitting a frame (when TBF = 0).

NOTE:
The transmit and receive buffers are not allowed to "wrap around" in the on-chip RAM. A "buffer end" is automatically generated if address 191 (BFH) is reached.

TCB: TRANSMIT CONTROL BYTE REGISTER (BYTE-ADDRESSABLE)

The Transmit Control Byte register (Address DAH) contains the byte which is to be placed in the control field of the transmitted frame, during NON-AUTO mode transmission. The CPU should access TCB only when the SIU is not transmitting a frame (when TBF = 0). The N_S and N_R counters are not used in the NON-AUTO mode.

RBS: RECEIVE BUFFER START ADDRESS REGISTER (BYTE-ADDRESSABLE)

The Receive Buffer Start address register (Address CCH) points to the location in on-chip RAM where the beginning of the I-field of the frame being received is to be stored. The CPU should write RBS only when the SIU is not receiving a frame (when RBE = 0).

RBL: RECEIVE BUFFER LENGTH REGISTER (BYTE-ADDRESSABLE)

The Receive Buffer Length register (Address CBH) contains the length (in bytes) of the area in on-chip RAM allocated for the received I-field. RBL = 0 is valid. The CPU should write RBL only when RBE = 0.

RFL: RECEIVE FIELD LENGTH REGISTER (BYTE-ADDRESSABLE)

The Received Field Length register (Address CDH) contains the length (in bytes) of the received I-field that has just been loaded into on-chip RAM. RFL is loaded by the SIU. RFL = 0 is valid. RFL should be accessed by the CPU only when RBE = 0.

RCB: RECEIVE CONTROL BYTE REGISTER (BYTE-ADDRESSABLE)

The Received Control Byte register (Address CAH) contains the control field of the frame that has just been received. RCB is loaded by the SIU. The CPU can only read RCB, and should only access RCB when RBE = 0.

7.3 ICE Support Registers

The 8044 In-Circuit Emulator (ICE-44) allows the user to exercise the 8044 application system and monitor the execution of instructions in real time.

The emulator operates with Intel's Intellec® development system. The development system interfaces with the user's 8044 system through an in-cable buffer box. The cable terminates in a 8044 pin-compatible plug, which fits into the 8044 socket in the user's system. With the emulator plug in place, the user can exercise his system in real time while collecting up to 255 instruction cycles of real-time data. In addition, he can single-step the program.

Static RAM is available (in the in-cable buffer box) to emulate the 8044 internal and external program memory and external data memory. The designer can display and alter the contents of the replacement memory in the buffer box, the internal data memory, and the internal 8044 registers, including the SFRs.

Among the SIU SFRs are the following registers that support the operation of the ICE:

DMA CNT: DMA COUNT REGISTER (BYTE-ADDRESSABLE)

The DMA Count register (Address CFH) indicates the number of bytes remaining in the information block that is currently being used.

FIFO: THREE-BYTE (BYTE-ADDRESSABLE)

The Three-Byte FIFO (Address DDH, DEH, and DFH) is used between the eight-bit shift register and the information buffer when an information block is received.

SIUST: SIU STATE COUNTER (BYTE-ADDRESSABLE)

The SIU State Counter (Address D9H) reflects the state of the internal logic which is under SIU control. Therefore, care must be taken not to write into this register.

The SIUST register can serve as a helpful aid to determine which field of a receive frame that the SIU expects next. The table below will help in debugging 8044 reception problems.

SIUST Value	Function
01H	Waiting for opening flag.
08H	Waiting for address field.
10H	Waiting for control field.
18H	Waiting for first byte of I field. This state is only entered if a FCS is expected. It pushes the received byte onto the top of the FIFO.
20H	Waiting for second byte of I field. This state always follows state 18H.
28H	Waiting for I field byte. This state can be entered from state 20H or from states 01H, 08H, or 10H depending upon the SIU's mode configuration. (Each time a byte is received, it is pushed onto the top of the FIFO and the byte at the bottom is put into memory. For no FCS formatted frames, the FIFO is collapsed into a single register).
30H	Waiting for the closing flag after having overflowed the receive buffer. Note that even if the receive frame overflows the assigned receive buffer length, the FCS is still checked.

Examples of SIUST status sequences for different frame formats are shown below. Note that status changes after acceptance of the received field byte.

Table 1. SIUST Status Sequences

										Frame Option		
										NFCS	NB	AM
Example 1:												
Frame Format	(Idle)	F	A	C	I			FCS	F	0	0	1
SIUST Value	01	01	08	10	18	20	28	28	01			
Example 2:												
Frame Format	(Idle)	F	A	I			FCS	F		0	1	1
SIUST Value	01	01	08	18	20	28	28	01				
Example 3:												
Frame Format	(Idle)	F	I			FCS	F			0	1	0
SIUST Value	01	01	18	20	28	28	01					
Example 4:												
Frame Format	(Idle)	F	A	I	F					1	1	1
SIUST Value	01	01	08	28	01							
Example 5:												
Frame Format	(Idle)	F	I		F					1	1	0
SIUST Value	01	01	28		01							
Example 6:												
Frame Format	(Idle)	F	I			I OVERFLOW		FCS	F	0	1	0
SIUST Value	01	01	18	20	28	30		30	01			

The SIU is initialized by a reset signal (on pin 9), followed by write operations to the SIU SFRs. Once initialized, the SIU can function in AUTO mode or NON-AUTO mode. Details are given below.

8.1 Initialization

Figure 6 is the SIU. Registers SMD, STS, and NSNR are cleared by reset. This puts the 8044 into an idle state—neither receiving nor transmitting. The following registers must be initialized before the 8044 leaves the idle state:

STAD — to establish the 8044's SDLC station address.

SMD — To configure the 8044 for the proper operating mode.

RBS, RBL — to define the area in RAM allocated for the Receive Buffer.

the Transmit Buffer.

Once these registers have been initialized, the user may write to the STS register to enable the SIU to leave the idle state, and to begin transmits and/or receives.

Setting RBE to 1 enables the SIU for receive. When RBE = 1, the SIU monitors the received data stream for a flag pattern. When a flag pattern is found, the SIU enters Receive mode and receives the frame.

Setting RTS to 1 enables the SIU for transmit. When RTS = 1, the SIU monitors the received data stream for a GA pattern (loop configuration) or waits for a CTS (non-loop configuration). When the GA or CTS arrives, the SIU enters Transmit mode and transmits a frame.

In AUTO mode, the SIU sets RTS to enable automatic transmissions of appropriate responses.

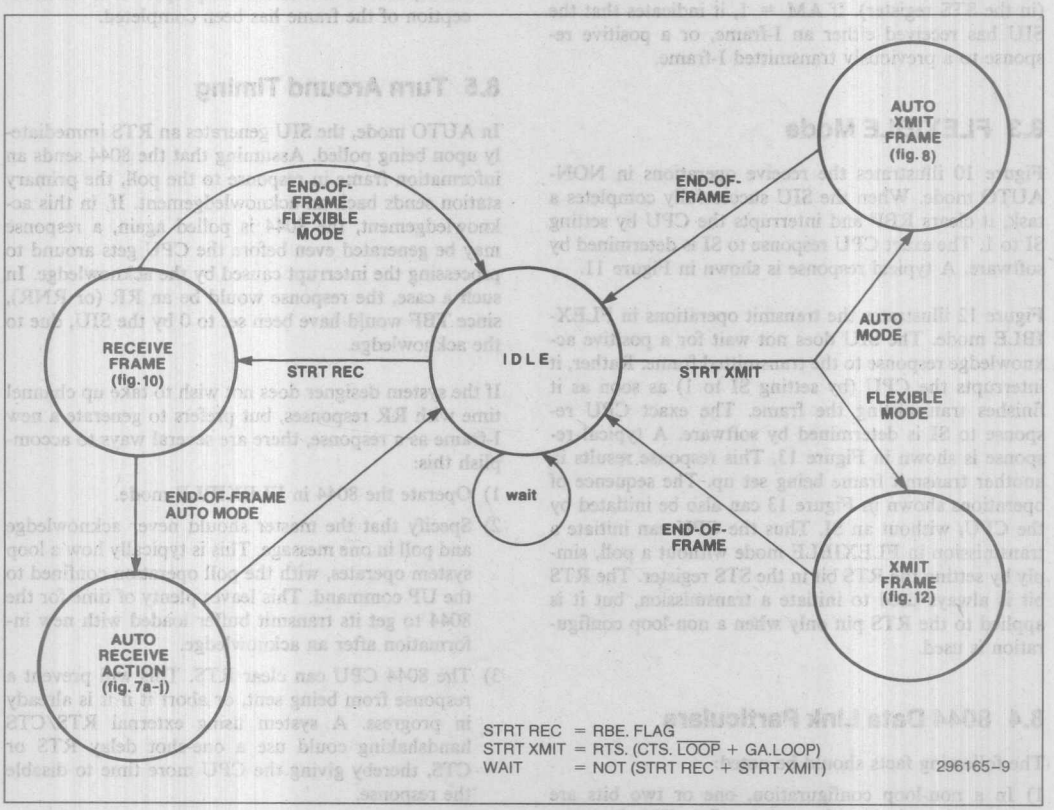


Figure 6. SIU State Diagram

8.2 AUTO Mode

Figure 7 illustrates the receive operations in AUTO mode. The overall operation is shown in Figure 7a. Particular cases are illustrated in Figures 7b through 7j. If any Unnumbered Command other than UP is received, the AM bit is cleared and the SIU responds as if in the FLEXIBLE mode, by interrupting the CPU for supervision. This will also happen if a BOV or SES condition occurs. If the received frame contains a poll, the SIU sets the RTS bit to generate a response.

Figure 8 illustrates the transmit operations in AUTO mode. When the SIU gets the opportunity to transmit, and if the transmit buffer is full, it sends an I-frame. Otherwise, it sends an RR if the buffer is free, or an RNR if the buffer is protected. The sequence counters NS and NR are used to construct the appropriate control fields.

Figure 9 shows how the CPU responds to an SI (serial interrupt) in AUTO mode. The CPU tests the AM bit (in the STS register). If AM = 1, it indicates that the SIU has received either an I-frame, or a positive response to a previously transmitted I-frame.

8.3 FLEXIBLE Mode

Figure 10 illustrates the receive operations in NON-AUTO mode. When the SIU successfully completes a task, it clears RBF and interrupts the CPU by setting SI to 1. The exact CPU response to SI is determined by software. A typical response is shown in Figure 11.

Figure 12 illustrates the transmit operations in FLEXIBLE mode. The SIU does not wait for a positive acknowledge response to the transmitted frame. Rather, it interrupts the CPU (by setting SI to 1) as soon as it finishes transmitting the frame. The exact CPU response to SI is determined by software. A typical response is shown in Figure 13. This response results in another transmit frame being set up. The sequence of operations shown in Figure 13 can also be initiated by the CPU, without an SI. Thus the CPU can initiate a transmission in FLEXIBLE mode without a poll, simply by setting the RTS bit in the STS register. The RTS bit is always used to initiate a transmission, but it is applied to the RTS pin only when a non-loop configuration is used.

8.4 8044 Data Link Particulars

The following facts should be noted:

1) In a non-loop configuration, one or two bits are transmitted before the opening flag. This is necessary for NRZI synchronization.
2) In a non-loop configuration, one to eight extra dribble bits are transmitted after the closing flag. These bits are a zero followed by ones.
3) In a loop configuration, when a GA is received and the 8044 begins transmitting, the sequence is 01111110101111110 ... (FLAG, 1, FLAG, ADDRESS, etc.). The first flag is created from the GA. The second flag begins the message.
4) CTS is sampled after the rising edge of the serial data, at about the center of the bit cell, except during a non-loop, externally clocked mode transmit, in which case it is sampled just after the falling edge.
5) The SIU does not check for illegal I-fields. In particular, if a supervisory command is received in AUTO mode, and if there is also an I-field, it will be loaded into the receive buffer (if RBP = 0), but it cannot cause a BOV.
6) In relation to the Receive Buffer Protect facility, the user should set RFL to 0 when clearing RBP, such that, if the SIU is in the process of receiving a frame, RFL will indicate the proper value when reception of the frame has been completed.

8.5 Turn Around Timing

In AUTO mode, the SIU generates an RTS immediately upon being polled. Assuming that the 8044 sends an information frame in response to the poll, the primary station sends back an acknowledgement. If, in this acknowledgement, the 8044 is polled again, a response may be generated even before the CPU gets around to processing the interrupt caused by the acknowledge. In such a case, the response would be an RR (or RNR), since TBF would have been set to 0 by the SIU, due to the acknowledge.

If the system designer does not wish to take up channel time with RR responses, but prefers to generate a new I-frame as a response, there are several ways to accomplish this:

1) Operate the 8044 in FLEXIBLE mode.
2) Specify that the master should never acknowledge and poll in one message. This is typically how a loop system operates, with the poll operation confined to the UP command. This leaves plenty of time for the 8044 to get its transmit buffer loaded with new information after an acknowledge.
3) The 8044 CPU can clear RTS. This will prevent a response from being sent, or abort it if it is already in progress. A system using external RTS/CTS handshaking could use a one-shot delay RTS or CTS, thereby giving the CPU more time to disable the response.

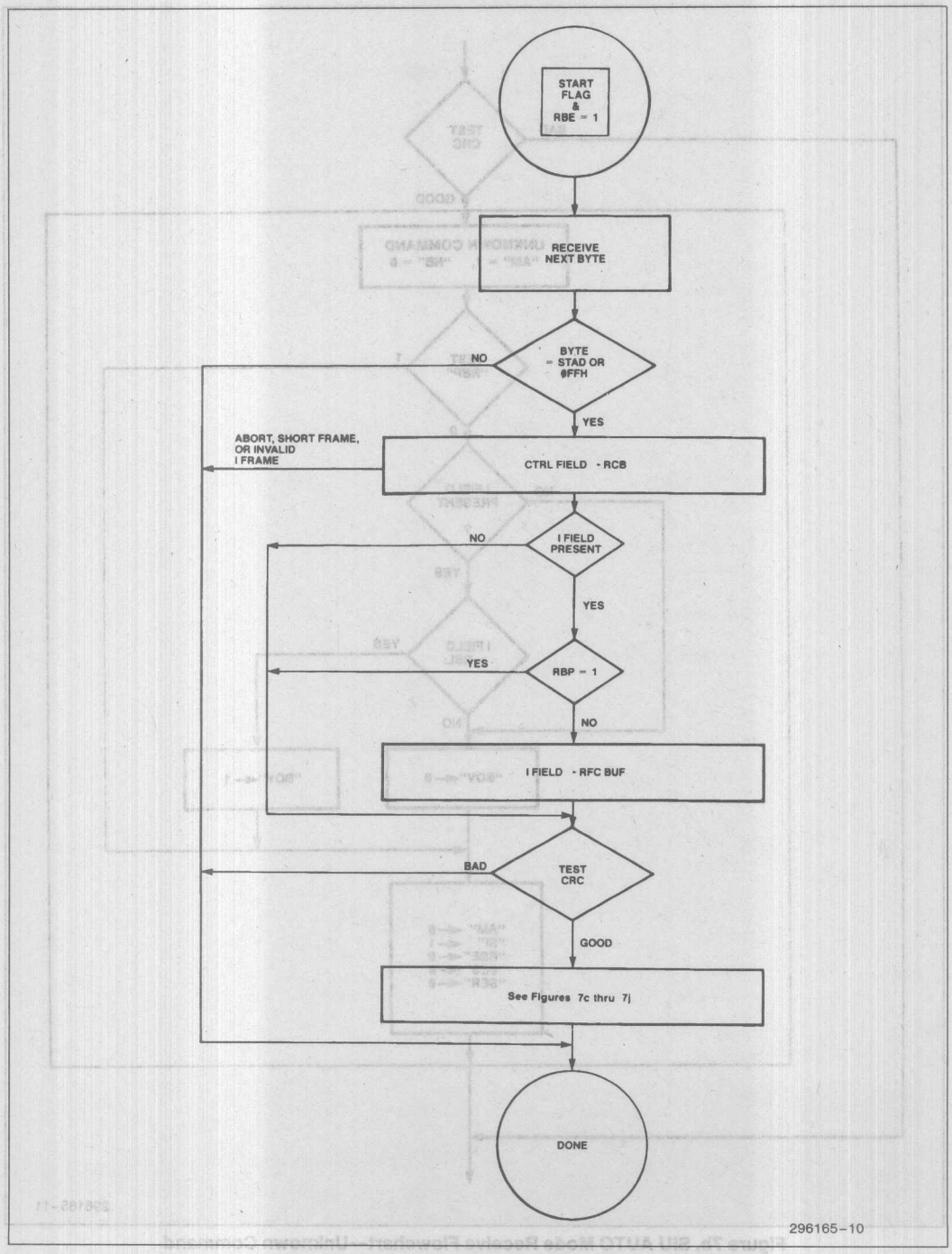


Figure 7a. SIU AUTO Mode Receive Flowchart—General

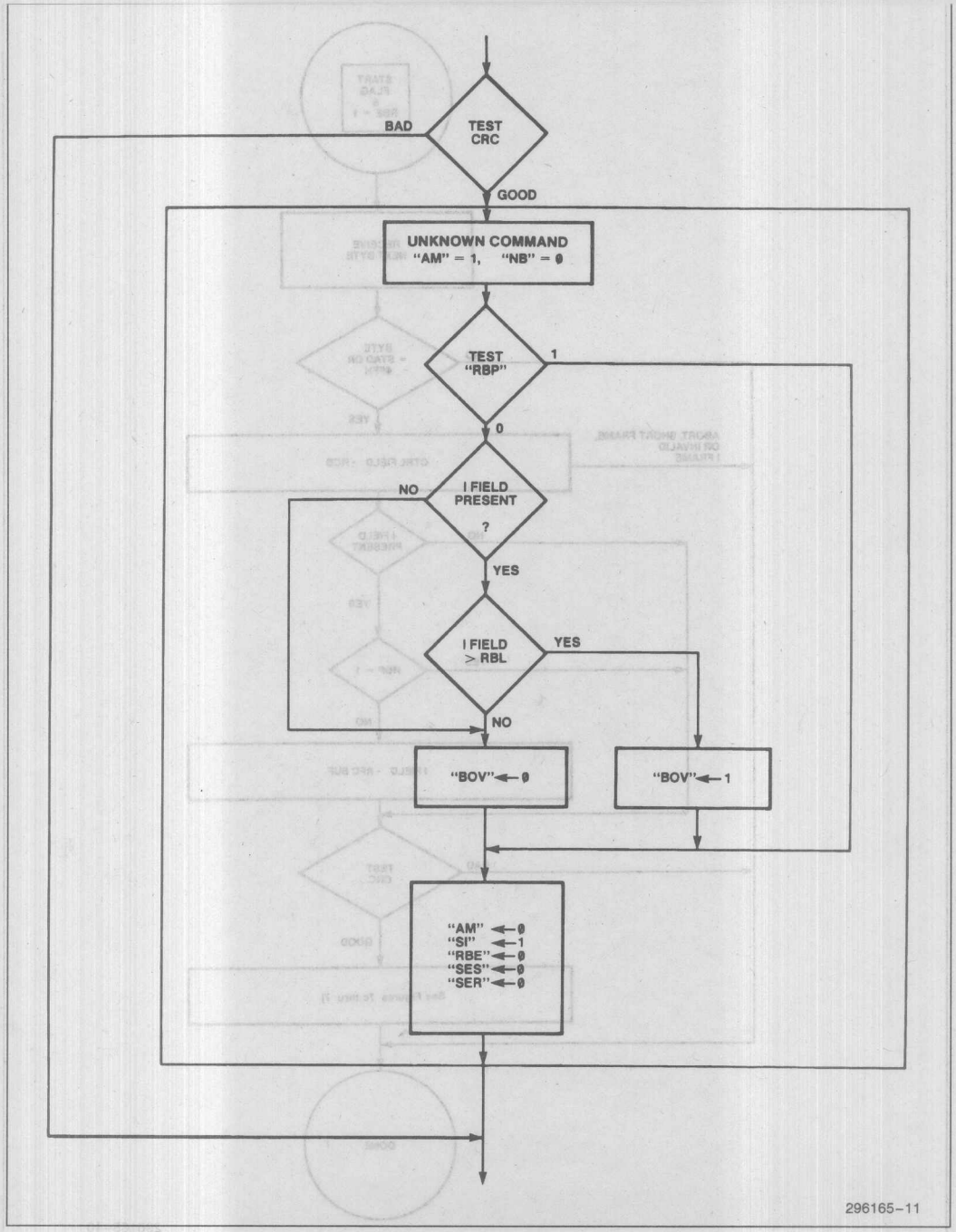


Figure 7b. SIU AUTO Mode Receive Flowchart—Unknown Command

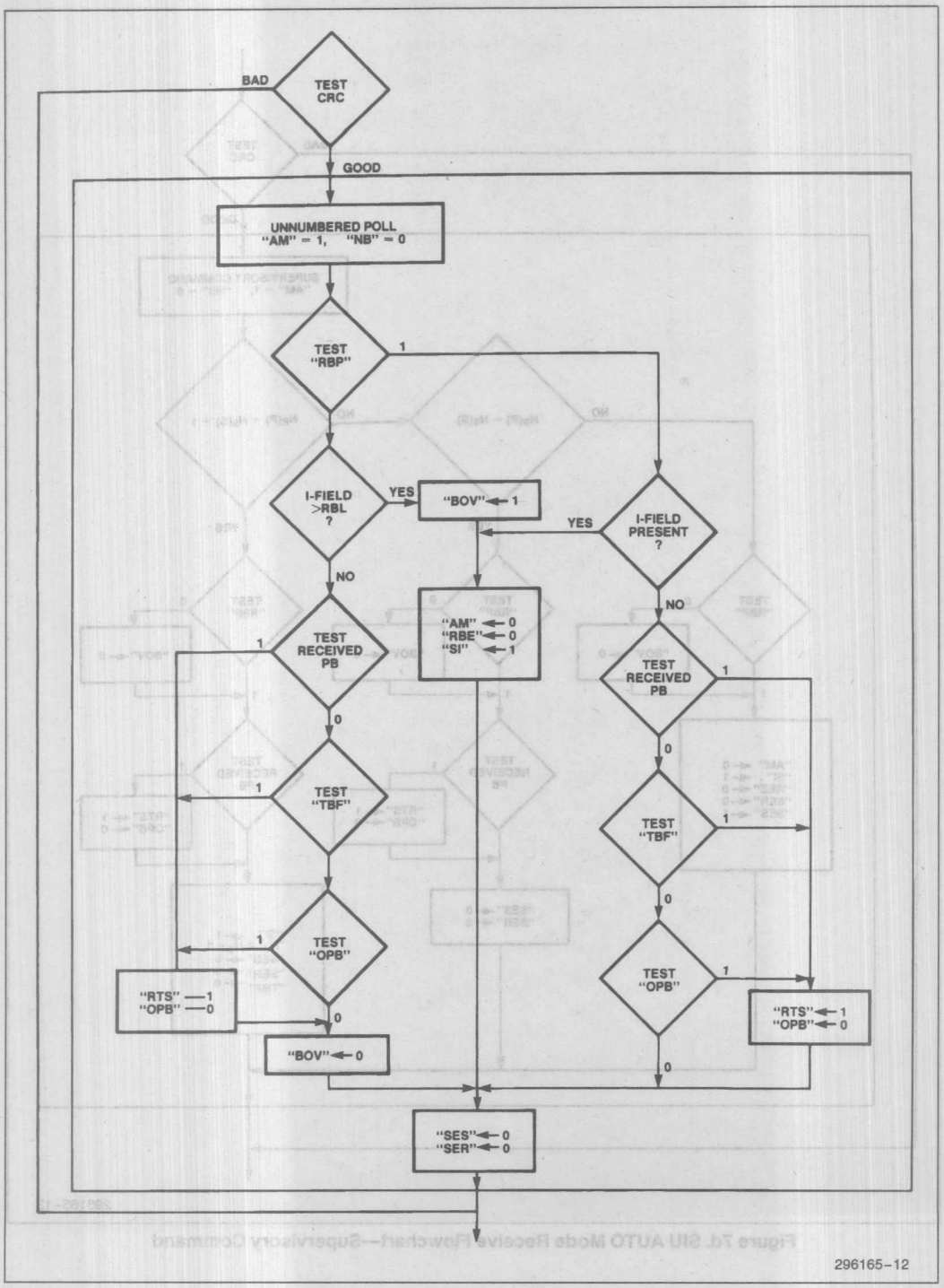


Figure 7c. SIU AUTO Mode Receive Flowchart—Unnumbered Poll

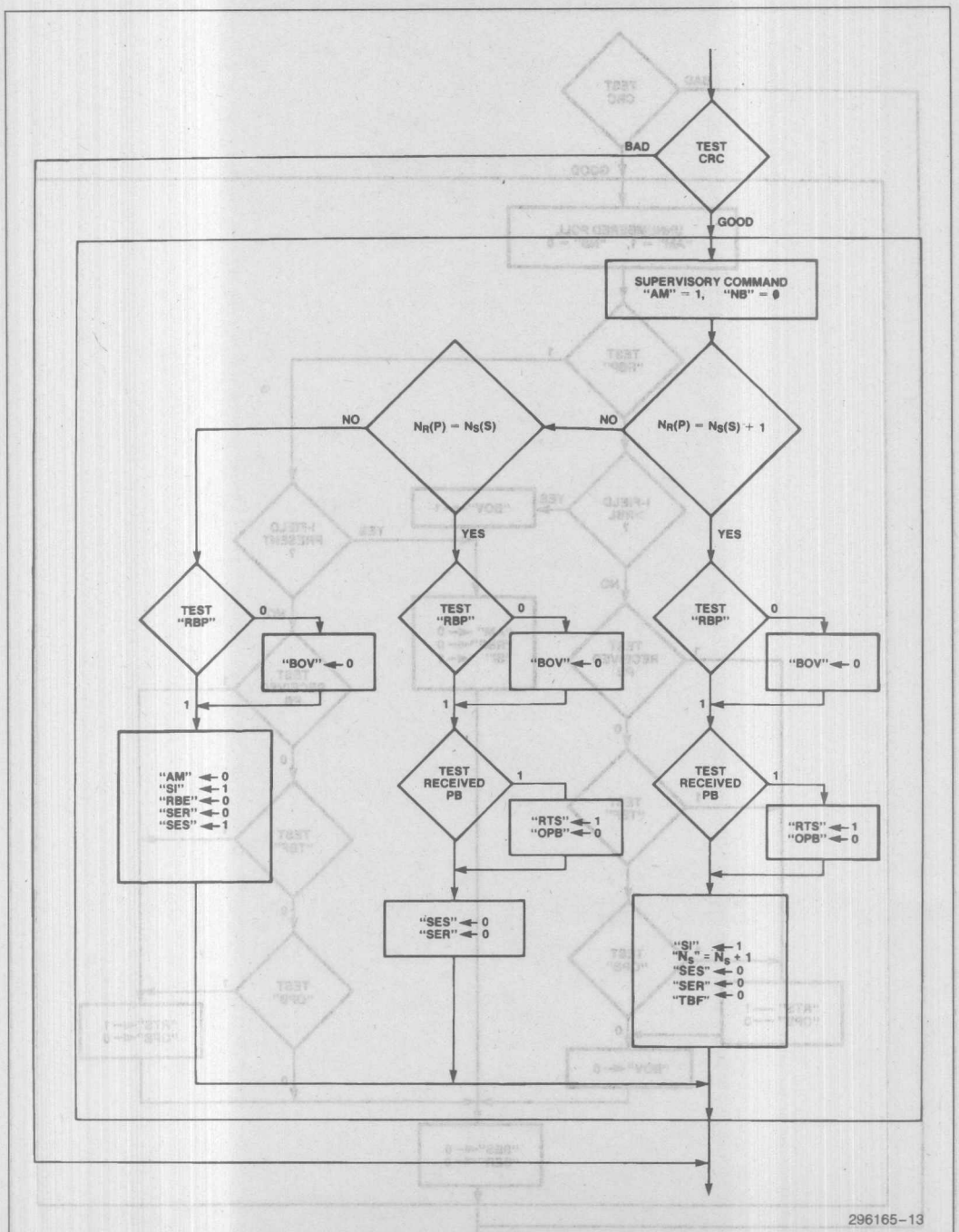


Figure 7d. SIU AUTO Mode Receive Flowchart—Supervisory Command

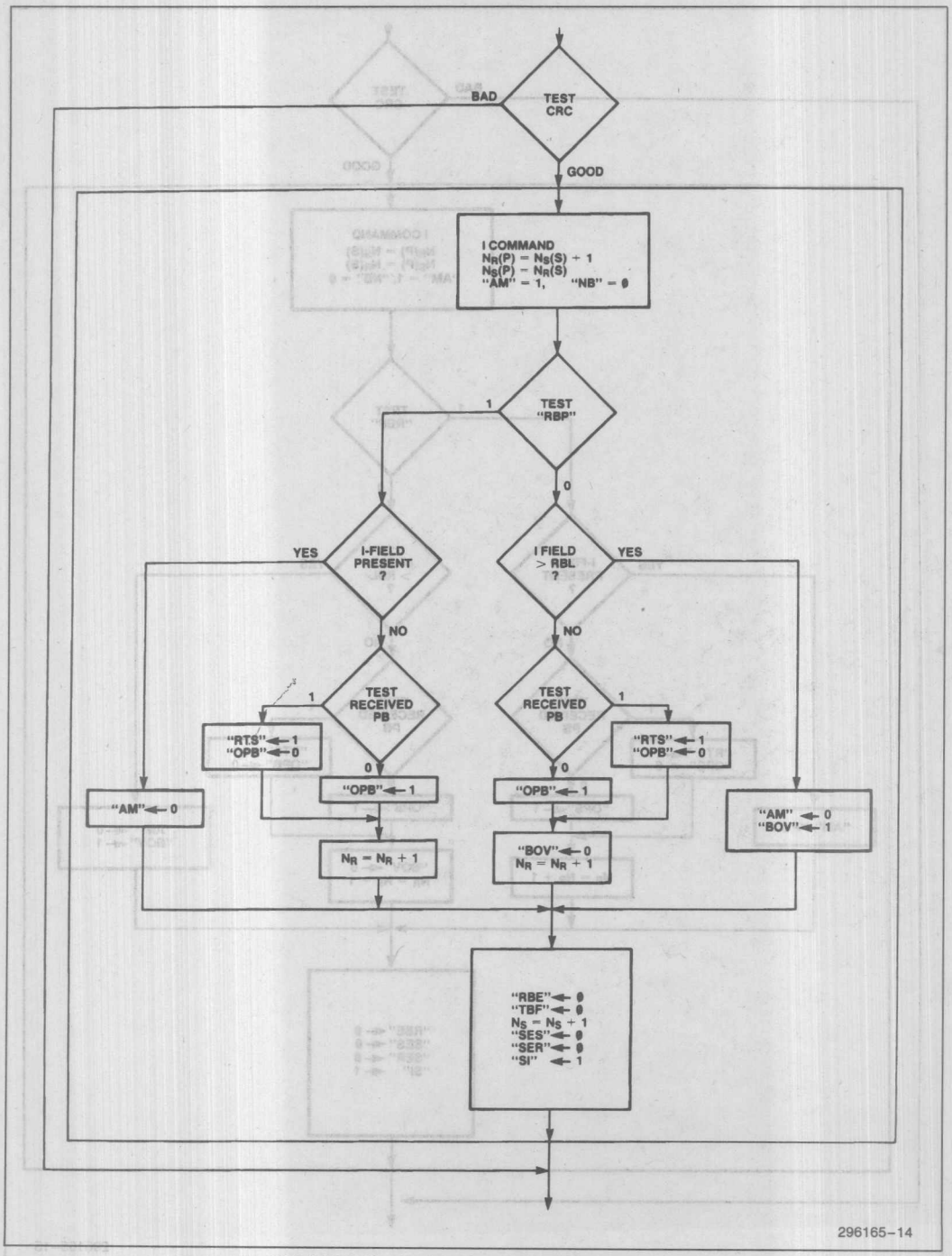


Figure 7e. SIU AUTO Mode Receive Flowchart—I Command: Prior Transmitted I-Field Confirmed, Current Received I-Field in Sequence

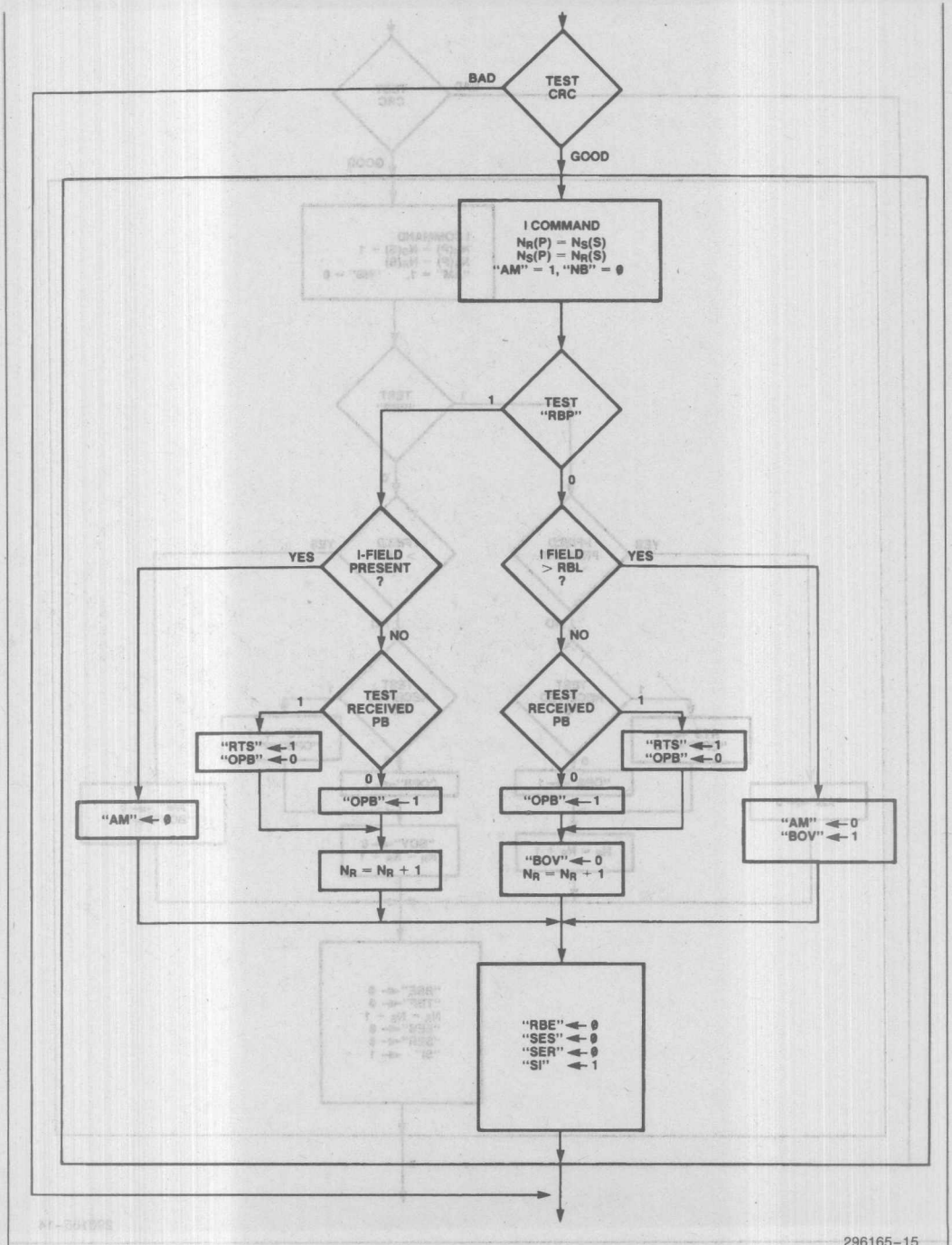


Figure 7f. SIU AUTO Mode Receive Flowchart—I Command: Prior Transmitted I-Field Not Confirmed, Current Received I-Field in Sequence

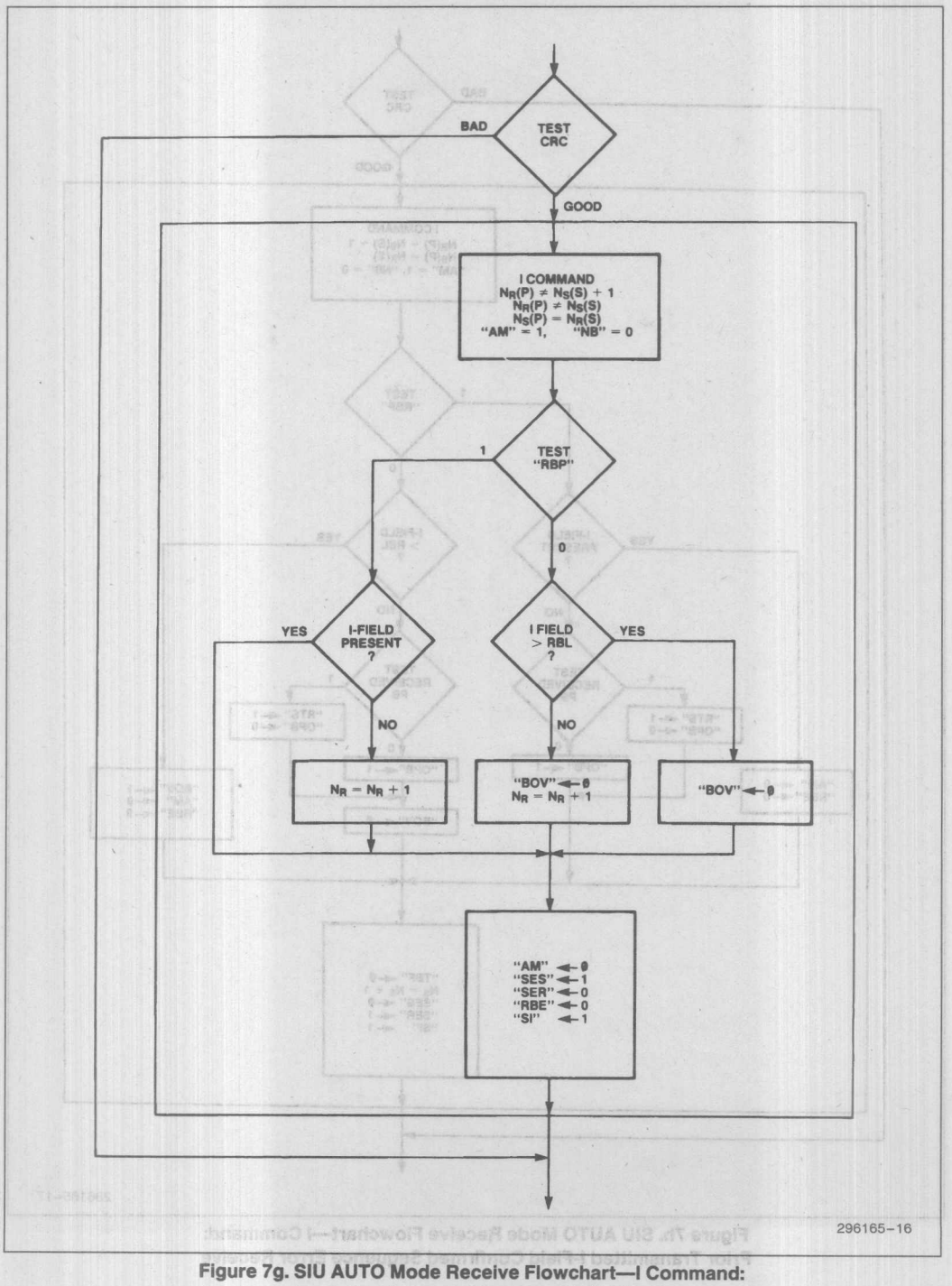


Figure 7g. SIU AUTO Mode Receive Flowchart—I Command: Sequence Error Send, Current Received I-Field in Sequence

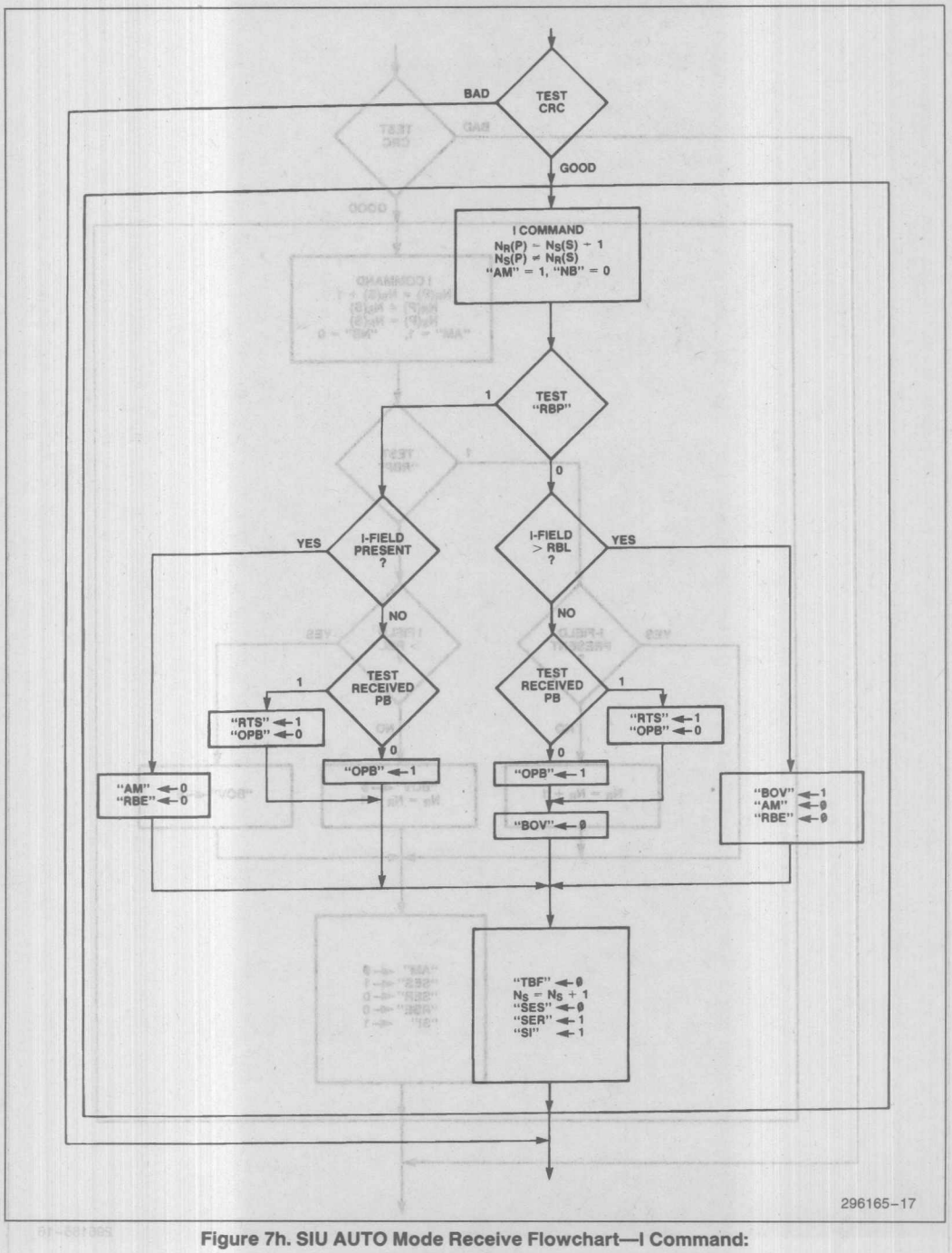


Figure 7h. SIU AUTO Mode Receive Flowchart—I Command: Prior Transmitted I-Field Confirmed Sequence Error Receive

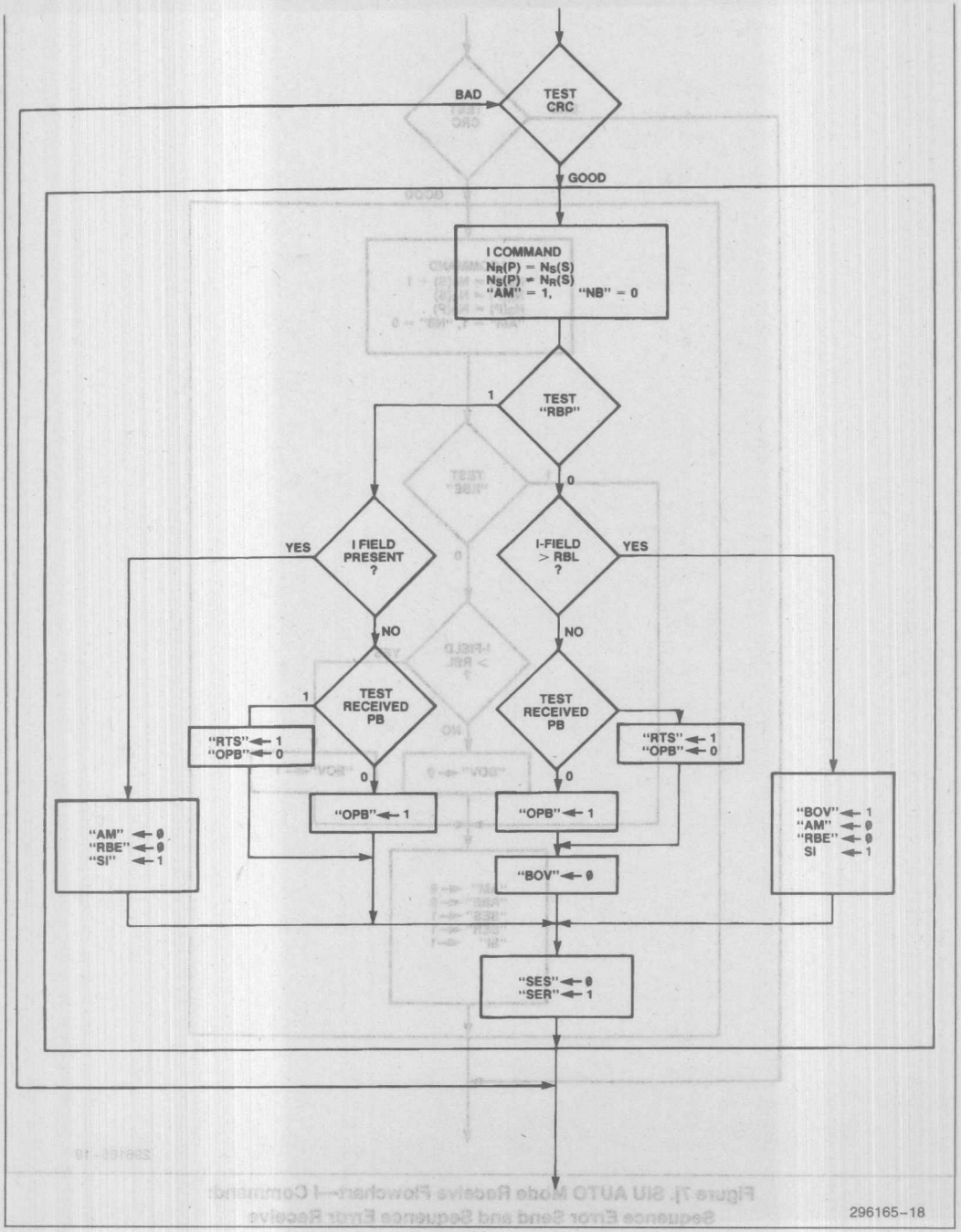


Figure 7i. SIU AUTO Mode Receive Flowchart—I Command: Prior Transmitted I-Field Not Confirmed, Sequence Error Receive

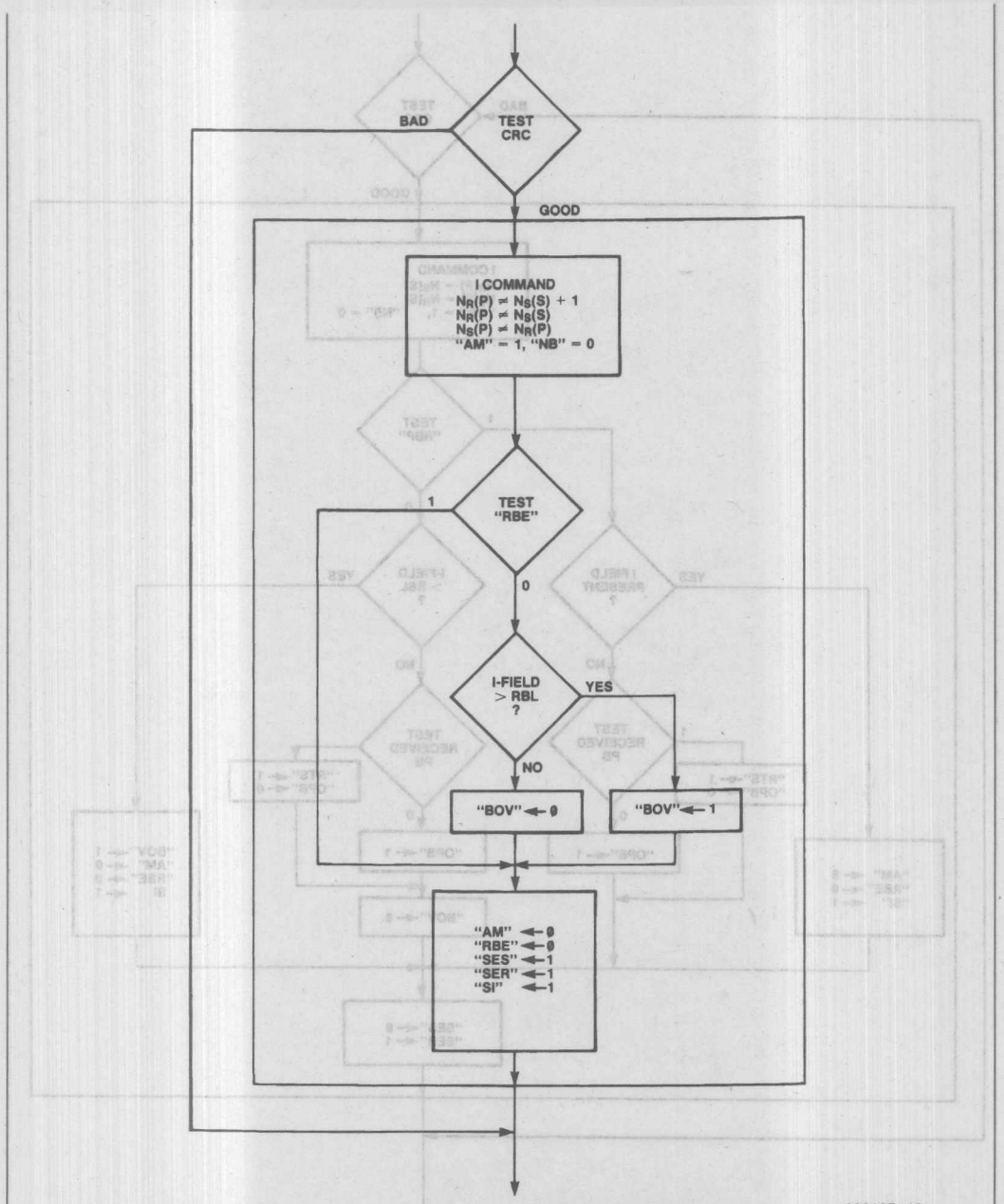


Figure 7j. SIU AUTO Mode Receive Flowchart—I Command: Sequence Error Send and Sequence Error Receive

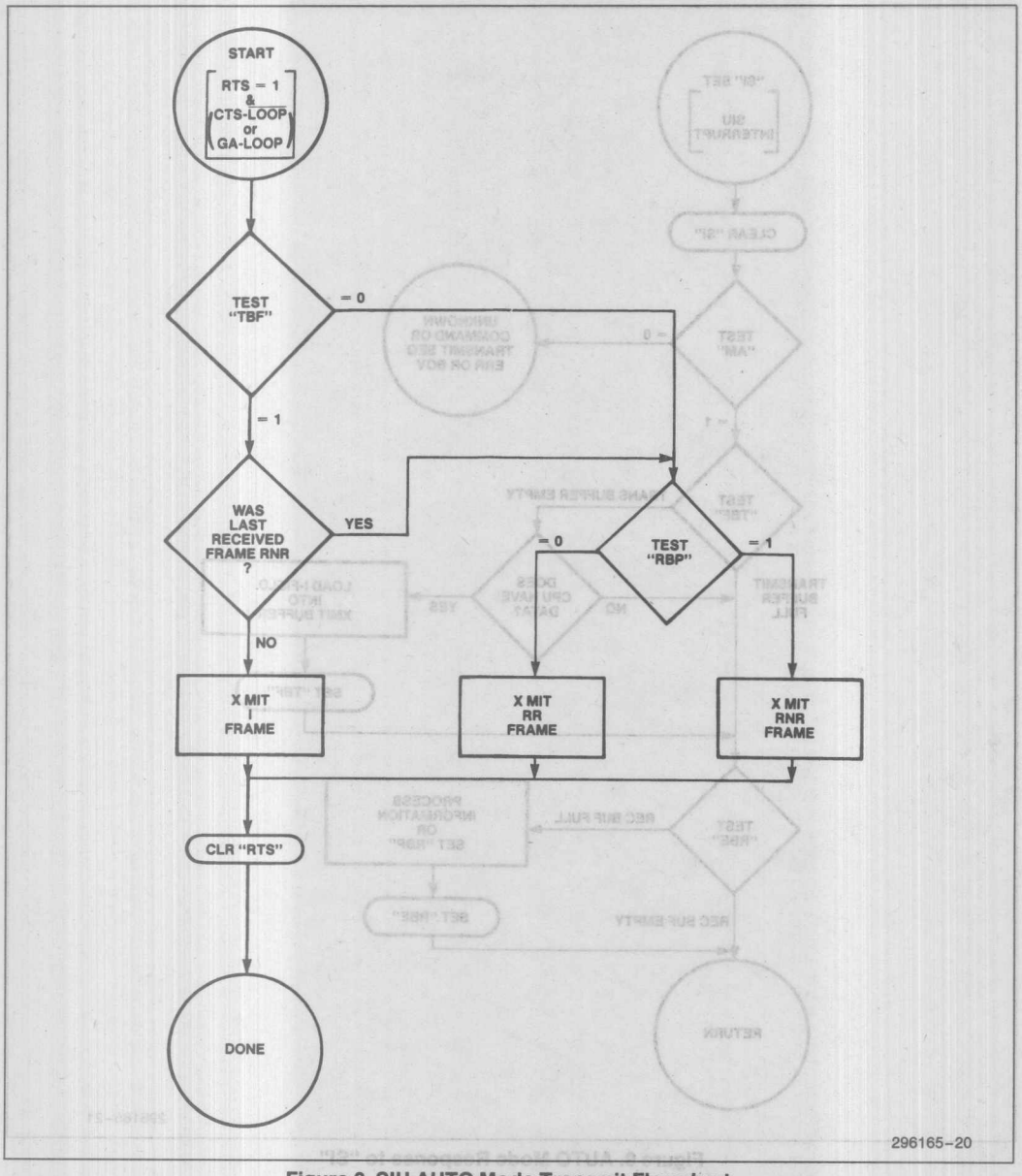


Figure 8. SIU AUTO Mode Transmit Flowchart

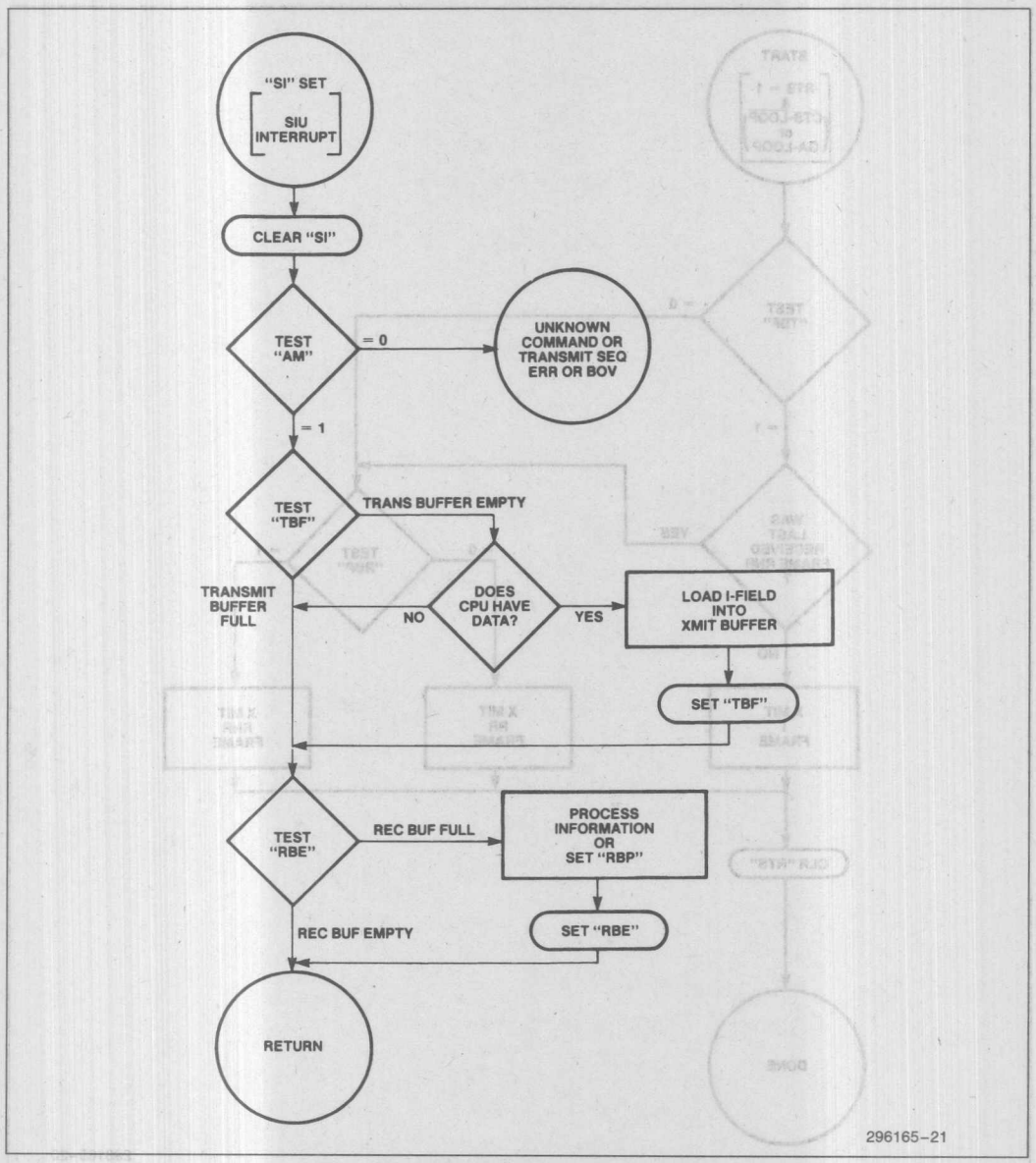


Figure 9. AUTO Mode Response to "SI"

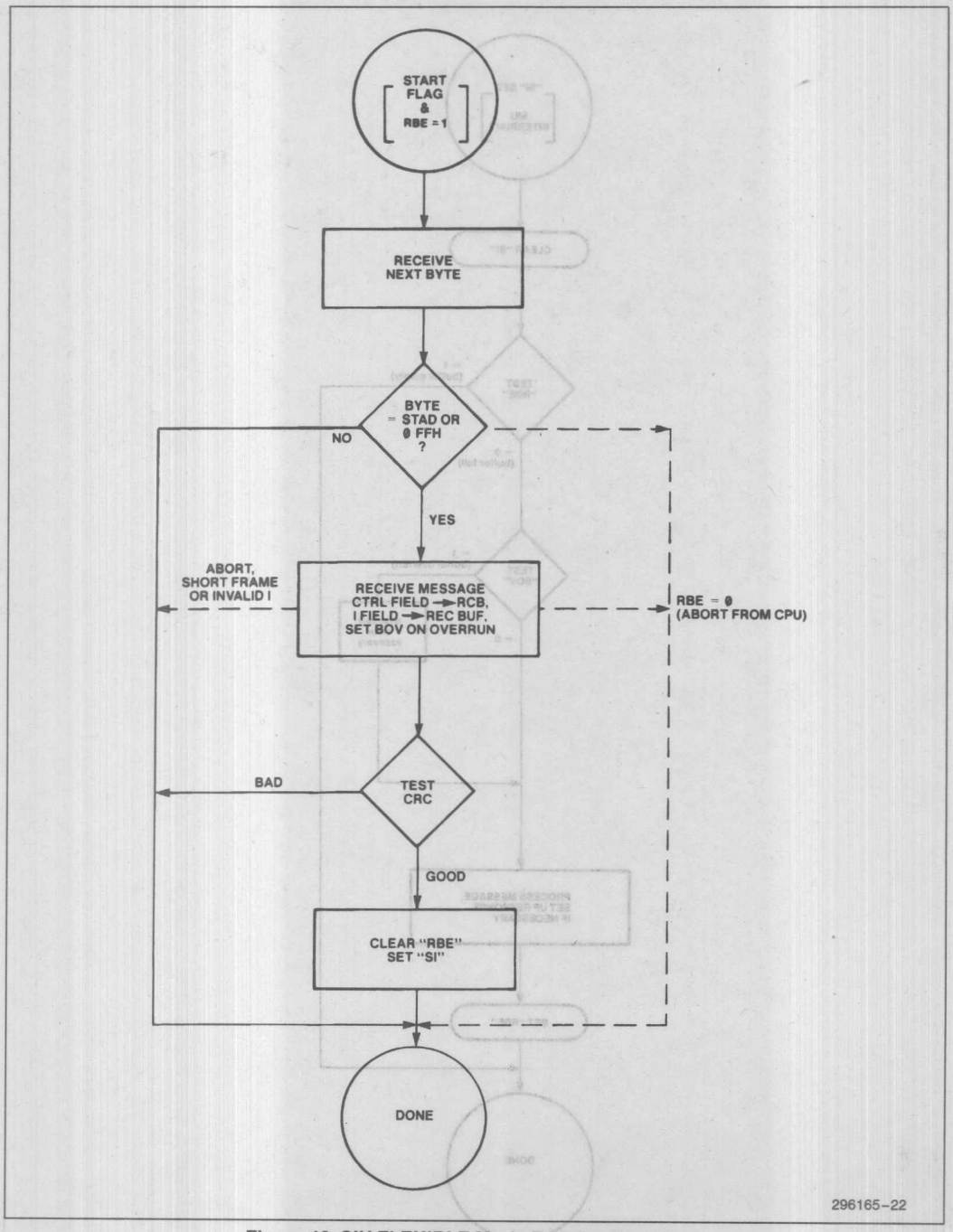


Figure 10. SIU FLEXIBLE Mode Receive Flowchart

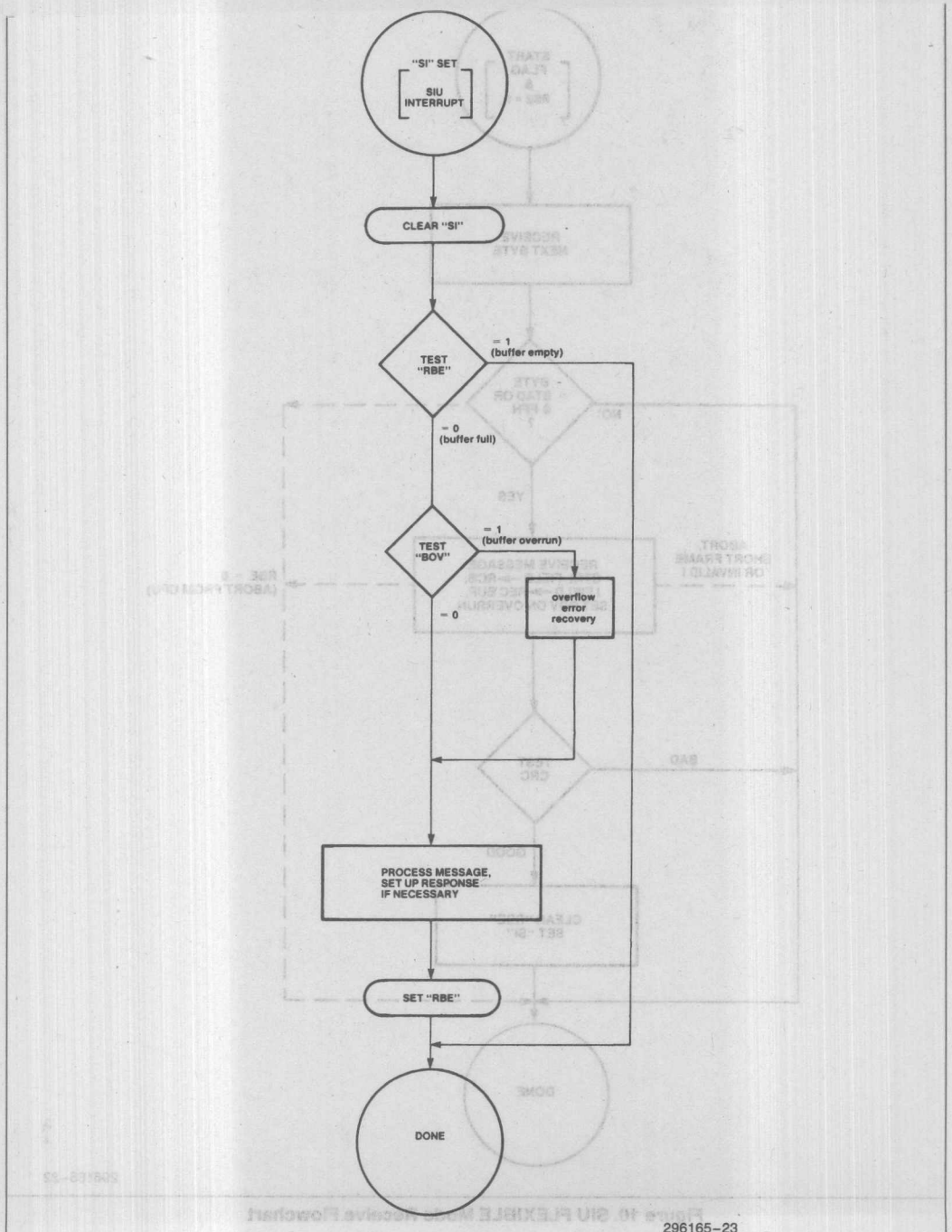


Figure 11. FLEXIBLE Mode Response to Receive "SI"

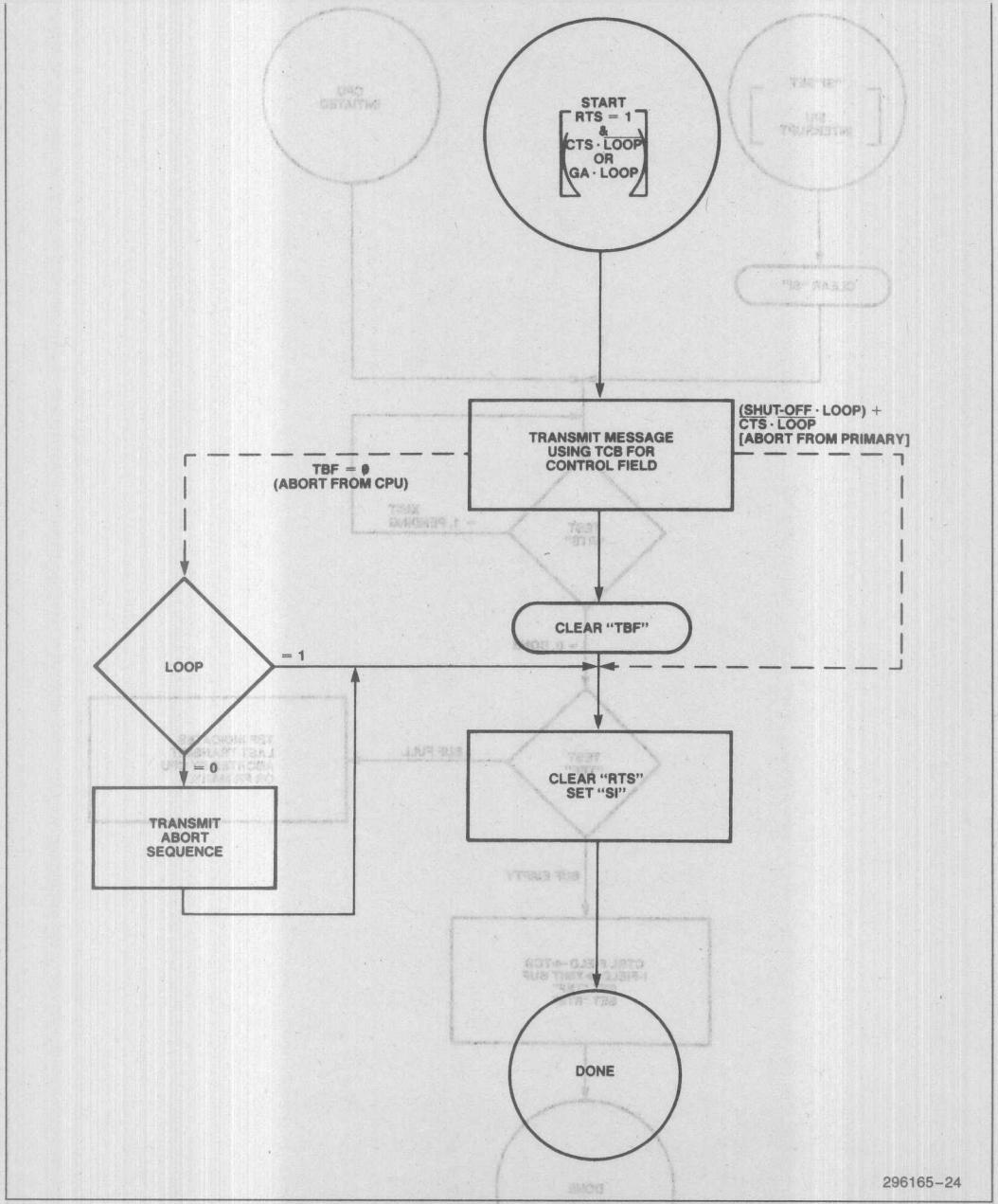


Figure 12. SIU FLEXIBLE Mode Transmit Flowchart

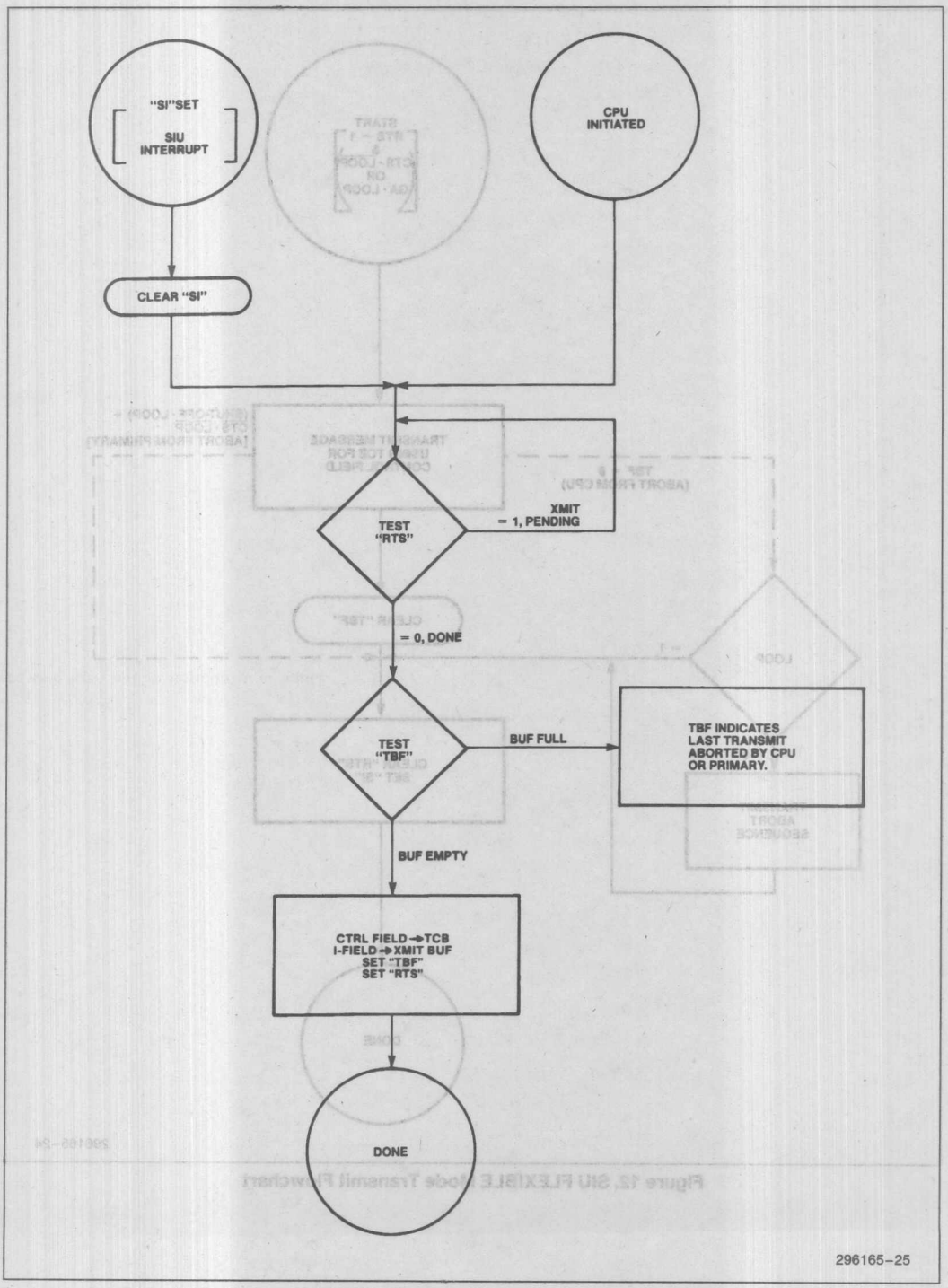


Figure 13. FLEXIBLE Mode Response to Transmit "SI"

9.0 MORE DETAILS ON SIU HARDWARE

The SIU divides functionally into two sections—a bit processor (BIP) and a byte processor (BYP)—sharing some common timing and control logic. As shown in Figure 14, the BIP operates between the serial port pins and the SIU bus, and performs all functions necessary to transmit/receive a byte of data to/from the serial data stream. These operations include shifting, NRZI encoding/decoding, zero insertion/deletion, and FCS generation/checking. The BYP manipulates bytes of data to perform message formatting, and other transmitting and receiving functions. It operates between the SIU bus (SIB) and the 8044's internal bus (IB). The interface between the SIU and the CPU involves an interrupt and some locations in on-chip RAM space which are managed by the BYP.

The maximum possible data rate for the serial port is limited to ½ the internal clock rate. This limit is imposed by both the maximum rate of DMA to the on-chip RAM, and by the requirements of synchronizing to an external clock. The internal clock rate for an 8044 running on a 12 MHz crystal is 6 MHz. Thus the maximum 8044 serial data rate is 3 MHz. This data rate drops down to 2.4 MHz when time is allowed for external clock synchronization.

9.1 The Bit Processor

In the asynchronous (self clocked) modes the clock is extracted from the data stream using the on-chip digital phase-locked-loop (DPLL). The DPLL requires a clock input at 16 times the data rate. This 16 × clock may originate from SCLK, Timer 1 Overflow, or PH2 (one half the oscillator frequency). The extra divide by-two described above allows these sources to be treated alternatively as 32 × clocks.

The DPLL is a free-running four-bit counter running off the 16 × clock. When a transition is detected in the receive data stream, a count is dropped (by suppressing the carry-in) if the current count value is greater than 8. A count is added (by injecting a carry into the second stage rather than the first) if the count is less than 8. No adjustment is made if the transition occurs at the count of 8. In this manner the counter locks in on the point at which transitions in the data stream occur at the count of 8, and a clock pulse is generated when the count overflows to 0.

In order to perform NRZI decoding, the NRZI decoder compares each bit of input data to the previous bit. There are no clock delays in going through the NRZI decoder.

The zero insert/delete circuitry (ZID) performs zero insertion/deletion, and also detects flags, GA's (Go-Ahead's), and aborts (same as GA's) in the data stream. The pattern 1111110 is detected as an early GA, so that the GA may be turned into a flag for loop mode transmission.

The shut-off detector monitors the receive data stream for a sequence of eight zeros, which is a shut-off command for loop mode transmissions. The shut-off detector is a three-bit counter which is cleared whenever a one is found in the receive data stream. Note that the ZID logic could not be used for this purpose, because the receive data must be monitored even when the ZID is being used for transmission.

As an example of the operation of the bit processor, the following sequence occurs in relation to the receive data:

1) RXD is sampled by SCLK, and then synchronized to the internal processor clock (IPC).
2) If the NRZI mode is selected, the incoming data is NRZI decoded.
3) When receiving other than the flag pattern, the ZID deletes the '0' after 5 consecutive '1's (during transmission this zero is inserted). The ZID locates the byte boundary for the rest of the circuitry. The ZID deletes the '0's by preventing the SR (shift register) from receiving a clocking pulse.
4) The FCS (which is a function of the data between the flags—not including the flags) is initialized and started at the detection of the byte boundary at the end of the opening flag. The FCS is computed each bit boundary until the closing flag is detected. Note that the received FCS has gone through the ZID during transmission.

9.2 The Byte Processor

Figure 15 is a block diagram of the byte processor (BYP). The BYP contains the registers and controllers necessary to perform the data manipulations associated with SDLC communications. The BYP registers may be read or written by the CPU over the 8044's internal bus (IB), using standard 8044 hardware register operations. The 8044 register select PLA controls these operations. Three of the BYP registers connect to the IB through the IBS, a sub-bus which also connects to the CPU interrupt control registers.

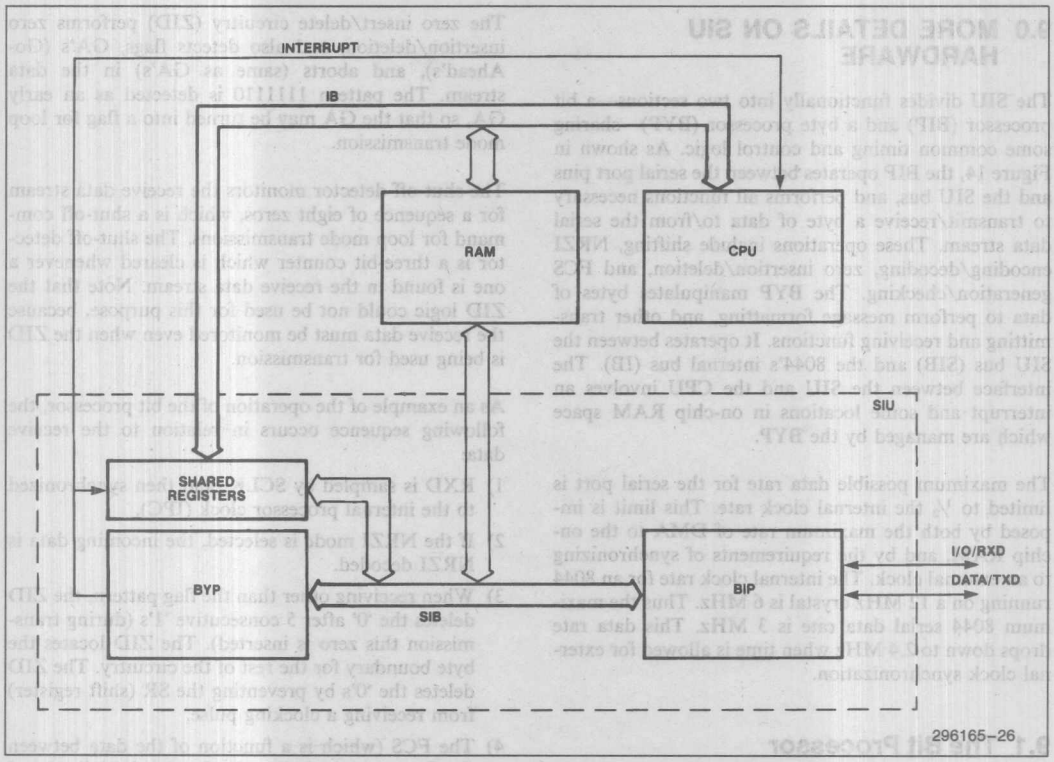


Figure 14. The Bit and Byte Processors

Simultaneous access of a register by both the IB and the SIB is prevented by timing. In particular, RAM access is restricted to alternate internal processor cycles for the CPU and the SIU, in such a way that collisions do not occur.

As an example of the operation of the byte processor, the following sequence occurs in relation to the receive data:

1) Assuming that there is an address field in the frame, the BYP takes the station address from the register file into temporary storage. After the opening flag, the next field (the address field) is compared to the station address in the temporary storage. If a match occurs, the operation continues.
2) Assuming that there is a control field in the frame, the BYP takes the next byte and loads it into the RCB register. The RCB register has the logic to update the NSNR register (increment receive count, set SES and SER flags, etc.).
3) Assuming that there is an information field, the next byte is dumped into RAM at the RBS location. The DMA CNT (RBL at the opening flag) is loaded from the DMA CNT register into the RB register and decremented. The RFL is then loaded into the RB register, incremented, and stored back into the register file.
4) This process continues until the DMA CNT reaches zero, or until a closing flag is received. Upon either event, the BYP updates the status, and, if the CRC is good, the NSNR register.

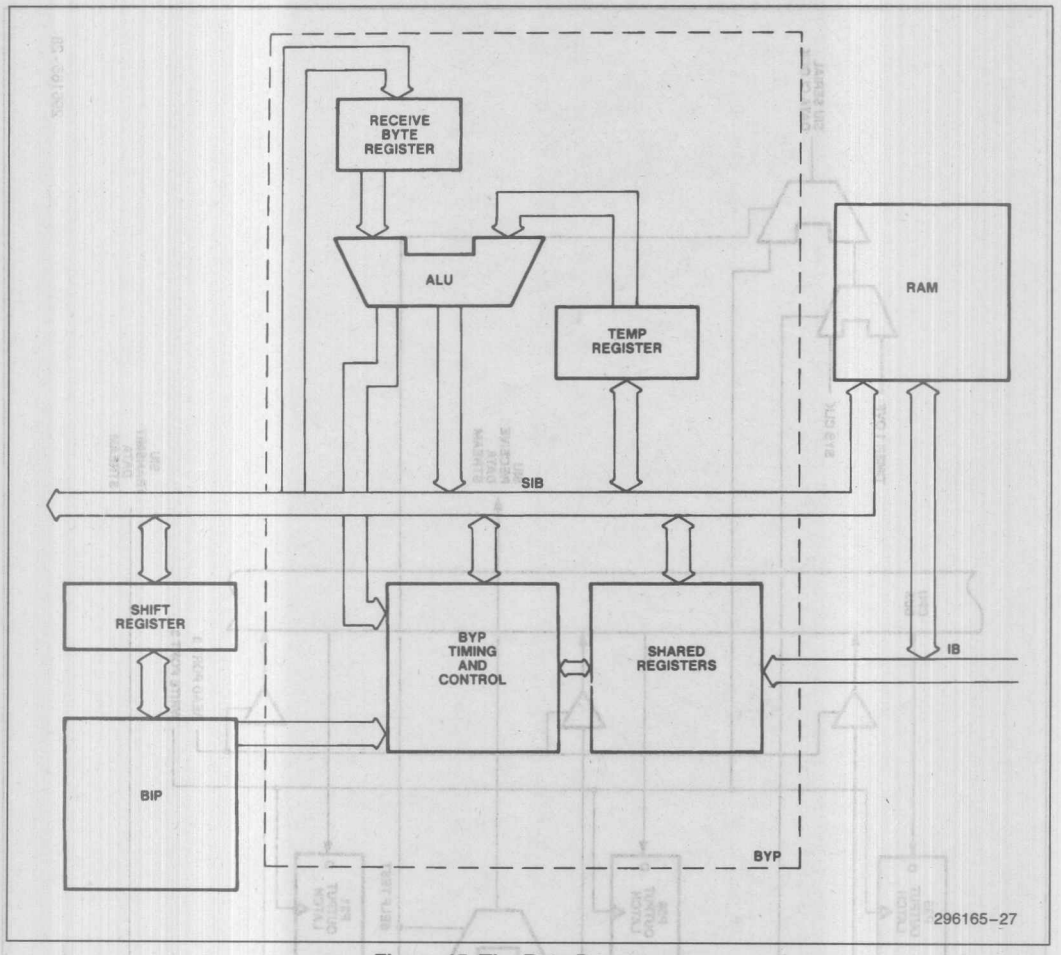


Figure 15. The Byte Processor

10.0 DIAGNOSTICS

An SIU test mode has been provided, so that the on-chip CPU can perform limited diagnostics on the SIU. The test mode utilizes the output latches for P3.0 and P3.1 (pins 10 and 11). These port 3 pins are not useful as out-put ports, since the pins are taken up by the serial port functions. Figure 16 shows the signal routing associated with the SIU test mode.

Writing a 0 to P3.1 enables the serial test mode (P3.1 is set to 1 by reset). In test mode the P3.0 bit is mapped into the received data stream, and the 'write port 3' control signal is mapped into the SCLK path in place of T1. Thus, in test mode, the CPU can send a serial data stream to the SIU by writing to P3.0. The transmit data stream can be monitored by reading P3.1. Each successive bit is transmitted from the SIU by writing to any bit in Port 3, which generates SCLK.

In test mode, the P3.0 and P3.1 pins are placed in a high voltage, high impedance state. When the CPU reads P3.0 and P3.1 the logic level applied to the pin will be returned. In the test mode, when the CPU reads 3.1, the transmit data value will be returned, not the voltage on the pin. The transmit data remains constant for a bit time. Writing to P3.0 will result in the signal being outputted for a short period of time. However, since the signal is not latched, P3.0 will quickly return to a high voltage, high impedance state.

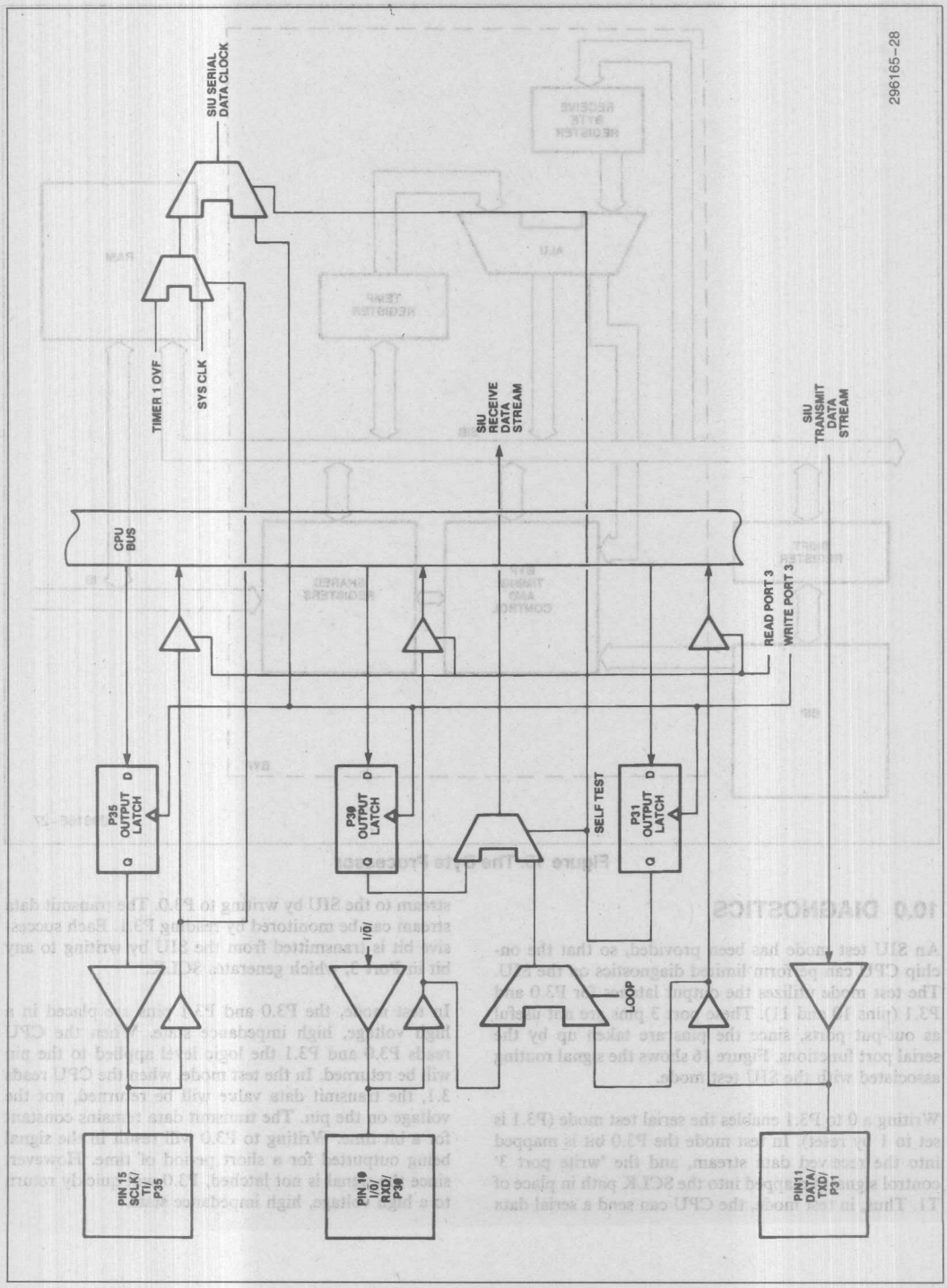


Figure 16. SIU Test Mode

The serial test mode is disabled by writing a 1 to P3.1. Care must be taken that a 0 is never written to P3.1 in the course of normal operation, since this causes the test mode to be entered.

Figure 17 is an example of a simple program segment that can be imbedded into the user's diagnostic program. That example shows how to put the 8044 into "Loop-back mode" to test the basic transmitting and receiving functions of the SIU.

Loop-back mode is functionally equivalent to a hard-wire connection between pins 10 and 11 on the 8044.

In this example, the 8044 CPU plays the role of the primary station. The SIU is in the AUTO mode. The CPU sends the SIU a supervisory frame with the poll bit set and an RNR command. The SIU responds with a supervisory frame with the poll bit set and an RR command.

The operation proceeds as follows:

Interrupts are disabled, and the self test mode is enabled by writing a zero to P3.1. This establishes P3.0 as the data path from the CPU to the SIU. CTS (clear-to-send) is enabled by writing a zero to P1.7. The station address is initialized by writing 08AH into the STAD (station address register).

The SIU is configured for receive operation in the clocked mode and in AUTO mode. The CPU then transmits a supervisory frame. This frame consists of an opening flag, followed by the station address, a control field indicating that this is a supervisory frame with an RNR command, and then a closing flag.

Each byte of the frame is transmitted by writing that byte into the A register and then calling the subroutine XMIT8. Two additional SCLKs are generated to guarantee that the last bits in the frame have been clocked into the SIU. Finally the CPU reads the status register (STS). If the operation has proceeded correctly, the status will be 072H. If it is not, the program jumps to the ERROR loop and terminates.

The SIU generates an SI (SIU interrupt) to indicate that it has received a frame. The CPU clears this interrupt, and then begins to monitor the data stream that is being generated by the SIU in response to what it has received. As each bit arrives (via P3.1), it is moved into the accumulator, and the CPU compares the byte in the accumulator with 07EH, which is the opening flag. When a match occurs, the CPU identifies this as byte boundary, and thereafter processes the information byte-to-byte.

The CPU calls the RCV8 subroutine to get each byte into the accumulator. The CPU performs compare operations on (successively) the station address, the control field (which contains the RR response), and the closing flag. If any of these do not compare, the program jumps to the ERROR loop. If no error is found, the program jumps to the DONE loop.

```
ISIS-II MCS-51 MACRO ASSEMBLER V2.0
OBJECT MODULE PLACED IN :F1:DATA.OBJ
ASSEMBLER INVOKED BY:  asm51 :f1:data.man device(44)


LOC  OBJ             LINE     SOURCE

                        1
                        2
0000 75C800             3     INIT:   MOV     STS,#00H
0003 C2B1               4             CLR     P3.1            ; Enable self test mode
0005 C297               5             CLR     P1.7            ; Enable CTS
0007 75CE8A             6             MOV     STAD, #8AH      ; Initialize address
                        7
                        8         ; CONFIGURE RECEIVE OPERATION
                        9
000A 75D86A            10             MOV     NSNR, #6AH      ; NS(S)=3, SES=0, NR(S)=5, SER=0
000D 75C901            11             MOV     SMD, #01H       ; NFCS=1
0010 75C8C2            12             MOV     STS, #0C2H      ; TBF=1, RBE=1, AM=1
                       13
                       14     ;   TRANSMIT A SUPERVISORY FRAME FROM THE PRIMARY STATION WITH THE POLL
                       15     ;   BIT SET AND A RNR COMMAND
                       16
0013 747E              17     SEND:   MOV     A, #7EH         ; The SIU receives a flag first
0015 120066            18             CALL    XMIT8
0018 748A              19             MOV     A, #8AH         ; The address is next
001A 120066            20             CALL    XMIT8
001D 7495              21             MOV     A, #095H        ; RNR SUP FRAME with P/F=1, NR(P)=4
001F 120066            22             CALL    XMIT8
0022 747E              23             MOV     A, #7EH         ; Receive closing flag
0024 120066            24             CALL    XMIT8
0027 D2B0              25             SETB    P3.0            ; Generate extra SCLK's to
0029 D2B0              26             SETB    P3.0            ; Initiate receive action
                       27
002B E5C8              28             MOV     A, STS          ; Check for appropriate status
002D B4722A            29             CJNE    A, #72H, ERROR
                       30
                       31     ; PREPARE TO RECEIVE RUPI'S RESPONCE TO PRIMARY'S RNR
                       32
                       33
                       34
0030 C2CC              35     RECV:   CLR     SI              ; Clear SI
0032 7400              36             MOV     A, #00H         ; Clear ACC
0034 780C              37             MOV     R3, #12         ; Try 12 times
                       38
                       39     ;  LOOK FOR THE OPENING FLAG
                       40
0036 D2B0              41     WFLAG1: SETB    P3.0            ; SCLK
0038 A2B1              42             MOV     C, P3.1         ; Transmitted data
003A 13                43             RRC     A
003B B47E03            44             CJNE    A, #07EH, WFLG1
003E 020046            45             JMP     CNTINU
0041 DBF3              46     WFLG1:  DJNZ    R3, WFLAG1
0043 02005A            47             JMP     ERROR
                       48
                       49
0046 12005C            50     CNTINU: CALL    RCVB            ; Get SIU's Transmitted address field
0049 B48A0E            51             CJNE    A, #08AH, ERROR
004C 12005C            52             CALL    RCVB            ; Primary expects to receive RR from SIU
004F B4B108            53             CJNE    A, #0B1H, ERROR
0052 12005C            54             CALL    RCVB            ; Receive closing flag
0055 B47E02            55             CJNE    A, #07EH, ERROR
                       56
0058 80FE              57     DONE:   JMP     DONE
                       58
005A 80FE              59     ERROR:  JMP     ERROR
                       60
                       61
005C 7808              62       RCVB: MOV     R0, #08         ; Initialize the bit counter
005E D2B0              63     GETBIT: SETB    P3.0            ; SCLK
0060 A2B1              64             MOV     C, P3.1         ; Transmitted data
0062 13                65             RRC     A
0063 D8F9              66             DJNZ    R0, GETBIT
0065 22                67             RET
                       68
                       69
                       70
0066 7809              71     XMIT8:  MOV     R0, #9          ; Initialize the bit counter
0068 13                72        L3:  RRC     A               ; Put the bit to be transmitted
                       73                                     ; in the Carry
0069 D801              74             DJNZ    R0, L1          ; When all bits have been sent
006B 22                75             RET                     ; return
                       76
006C 4004              77          L1: JC      L2              ; If the carry bit is set, set
                       78                                     ; port P3.0 else
006E C2B0              79             CLR     P3.0            ; clear port P3.0
0070 80F6              80             JMP     L3
                       81
0072 D2B0              82          L2: SETB    P3.0
0074 80F2              83             JMP     L3
                       84     end
```

296165-29

Figure 17. Loop-Back Mode Software

8044 Application Examples 14

1.0 INTERFACING THE 8044 TO A MICROPROCESSOR

The 8044 is designed to serve as an intelligent controller for remote peripherals. However, it can also be used as an intelligent HDLC/SDLC front end for a microprocessor, capable of extensively off-loading link control functions for the CPU. In some applications, the 8044 can even be used for communications preprocessing, in addition to data link control.

This section describes a sample hardware interface for attaching the 8044 to an 8088. It is general enough to be extended to other microprocessors such as the 8086 or the 80186.

OVERVIEW

A sample interface is shown in Figure 1. Transmission occurs when the 8088 loads a 64 byte block of memory with some known data. The 8088 then enables the 8237A to DMA this data to the 8044. When the 8044 has received all of the data from the 8237A, it sends the data in a SDLC frame. The frame is captured by the Spectron Datascope™* which displays it on a CRT in hex format.

In reception, the Datascope sends a SDLC information frame to the 8044. The 8044 receives the SDLC frame, buffers it, and sends it to the 8088's memory. In this example the 8044 is being operated in the NON-AUTO mode; therefore, it does not need to be polled by a primary station in order to transmit.

THE INTERFACE

The 8044 does not have a parallel slave port. The 8044's 32 I/O lines can be configured as a local microprocessor bus master. In this configuration, the 8044 can expand the ROM and RAM memory, control peripherals, and communicate with a microprocessor.

The 8044, like the 8051, does not have a Ready line, so there is no way to put the 8044 in wait state. The clock on the 8044 cannot be stopped. Dual port RAM could still be used, however, software arbitration would be the only way to prevent collisions. Another way to interface the 8044 with another CPU is to put a FIFO or queue between the two processors, and this was the method chosen for this design.

Figure 2 shows the schematic of the 8044/8088 interface. It involves two 8-bit tri-state latches, two SR flip-flops, and some logic gates (6 TTL packs). The circuitry implements a one byte FIFO. RS422 transceivers are used, which can be connected to a multidrop link. Figure 3 shows the 8088 and support circuitry; the memory and decoders are not shown. It is a basic 8088 Min Mode system with an 8237A DMA controller and an 8259A interrupt controller.

DMA Channel One transfers a block of memory to the tri-state latch, while Channel Zero transfers a block of data from the latch to 8088's memory. The 8044's Interrupt 0 signal vectors the CPU into a routine which reads from the internal RAM and writes to the latch. The 8044's Interrupt 1 signal causes the chip to read from the latch and write to its on-chip data RAM. Both DMA requests and acknowledges are active low.

Initially, when the power is applied, a reset pulse coming from the 8284A initializes the SR flip-flops. In this initialization state, the 8044's transmit interrupt and the 8088's transmit DMA request are active; however, the software keeps these signals disabled until either of the two processors are ready to transmit. The software leaves the receive signals enabled, unless the receive buffers are full. In this way either the 8088 or the 8044 are always ready to receive, but they must enable the transmit signal when they have prepared a block to transmit. After a block has been transmitted or received, the DMA and interrupt signals return to the initial state.

The receive and transmit buffer sizes for the blocks of data sent between the 8044 and the 8088 have a maximum fixed length. In this case the buffer size was 64 bytes. The buffer size must be less than 192 bytes to enable 8044 to buffer the data in its on-chip RAM. This design allows blocks of data that are less than 64 bytes, and accommodates networks that allow frames of varying size. The first byte transferred between the 8088 and the 8044 is the byte count to follow; thus the 8044 knows how many bytes to receive before it transmits the SDLC frame. However, when the 8044 sends data to the 8088's memory, the 8237A will not know if the 8044 will send less than the count the 8237A was programmed for. To solve this problem, the 8237A is operated in the single mode. The 8044 uses an I/O bit to generate an interrupt request to the 8259A. In the 8088's interrupt routine, the 8237A's receive DMA channel is disabled, thus allowing blocks of data less than 64 bytes to be received.

THE SOFTWARE

The software for the 8044 and the 8088 is shown in Table 1. The 8088 software was written in PL/M86, and the 8044 software was written in assembly language.

The 8044 software begins by initializing the stack, interrupt priorities, and triggering types for the interrupts. At this point, the SIU parameter registers are

*Datascope is a trademark of Spectron Inc.

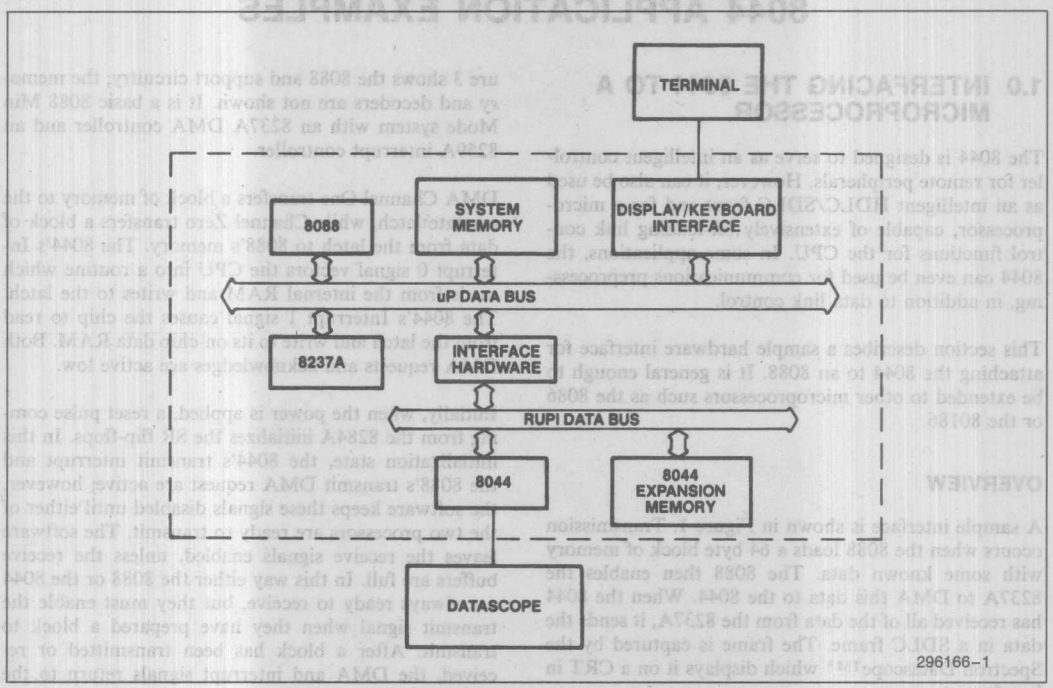


Figure 1. Block Diagram of 8088/8044 Interface Test

initialized. The receive and transmit buffer starting addresses and lengths are loaded for the on-chip DMA. This DMA is for the serial port. The serial station address and the transmit control bytes are loaded too.

Once the initialization has taken place, the SIU interrupt is enabled, and the external interrupt which receives bytes from the 8088 is enabled. Setting the 8044's Receive Buffer Empty (RBE) bit enables the receiver. If this bit is reset, no serial data can be received. The 8044 then waits in a loop for either RECEIVE DMA interrupt or the SERIAL INT interrupt.

The RECEIVE DMA interrupt occurs when the 8237A is transferring a block of data to the 8044. The first time this interrupt occurs, the 8044 reads the latch and loads the count value into the R2 register. On subsequent interrupts, the 8044 reads the latch, loads the data into the transmit buffer, and decrements R2. When R2 reaches zero, the interrupt routine sends the data in an SDLC frame, and disables the RECEIVE DMA interrupt. After the frame has been transmitted, a serial interrupt is generated. The SERIAL INT routine detects that a frame has been transmitted and re-enables the RECEIVE DMA interrupt. Thus, while the frame is being transmitted through the SIU, the 8237A is inhibited from sending data to the 8044's transmit buffer.

The TRANSMIT DMA routine sends a block of data from the 8044's receive buffer to the 8088's memory. Normally this interrupt remains disabled. However, if a serial interrupt occurs, and the SERIAL INT routine detects that a frame has been received, it calls the SEND subroutine. The SEND subroutine loads the number of bytes which were received in the frame into the receive buffer. Register R1 points to the receive buffer and R2 is loaded with the count. The TRANSMIT DMA interrupt is enabled, and immediately upon returning from the SERIAL INT routine, the interrupt is acknowledged. Each time the TRANSMIT DMA interrupt occurs, a byte is read from the receive buffer, written to the latch, and R2 is decremented. When R2 reaches 0, the TRANSMIT DMA interrupt is disabled, the SIU receiver is re-enabled, and the 8044 interrupts the 8088.

CONCLUSION

For the software shown in Table 1, the transfer rate from the 8088's memory to the 8044 was measured at 75K bytes/sec. This transfer rate largely depends upon the number of instructions in the 8044's interrupt service routine. Fewer instructions result in a higher transfer rate.

There are many ways of interfacing the 8044 locally to another microprocessor: FIFO's, dual port RAM with software arbitration, and 8255's are just a few. Alternative approaches, which may be more optimal for certain applications, are certainly possible.

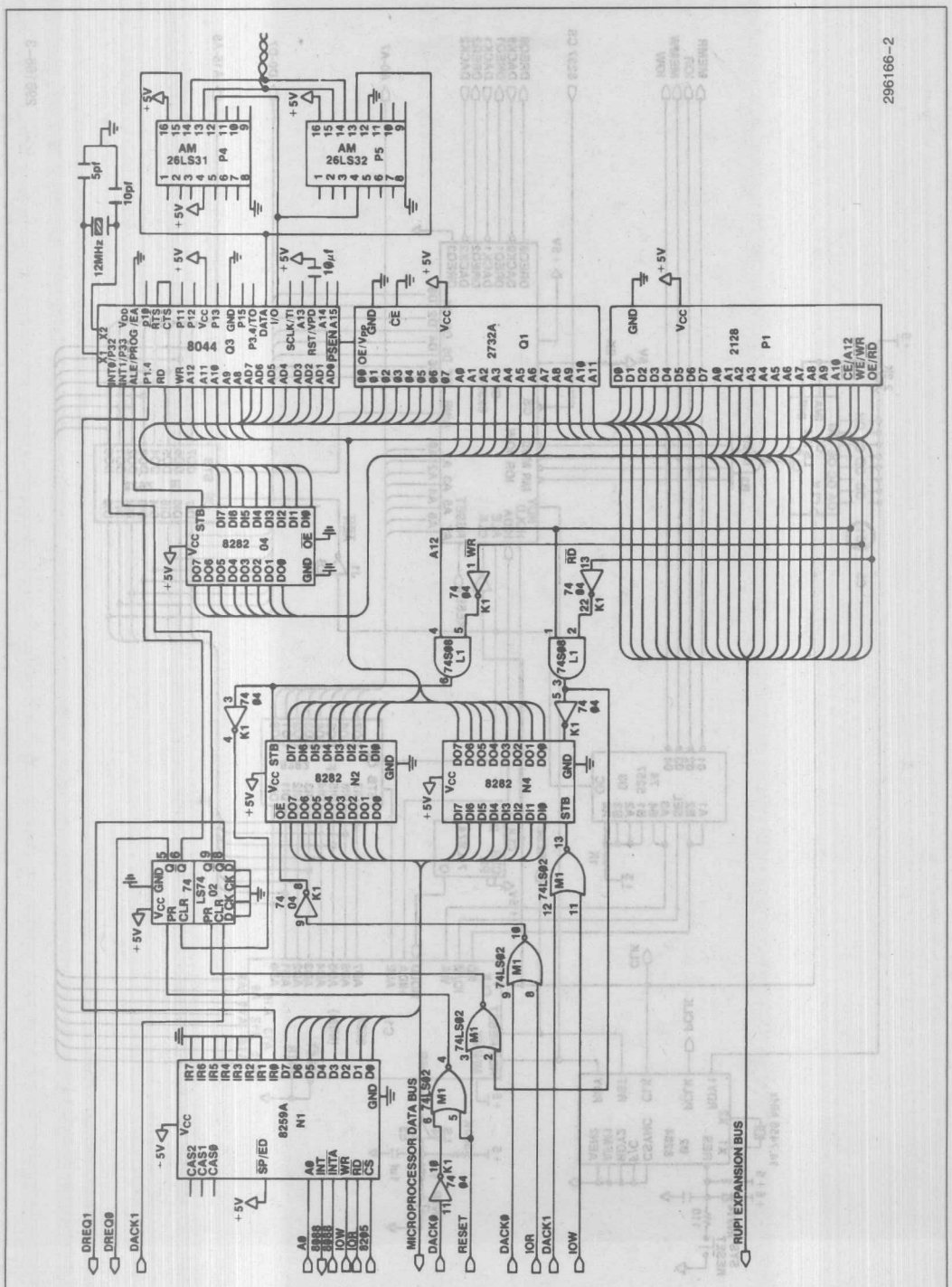


Figure 2. 8044 Interface to the 8088

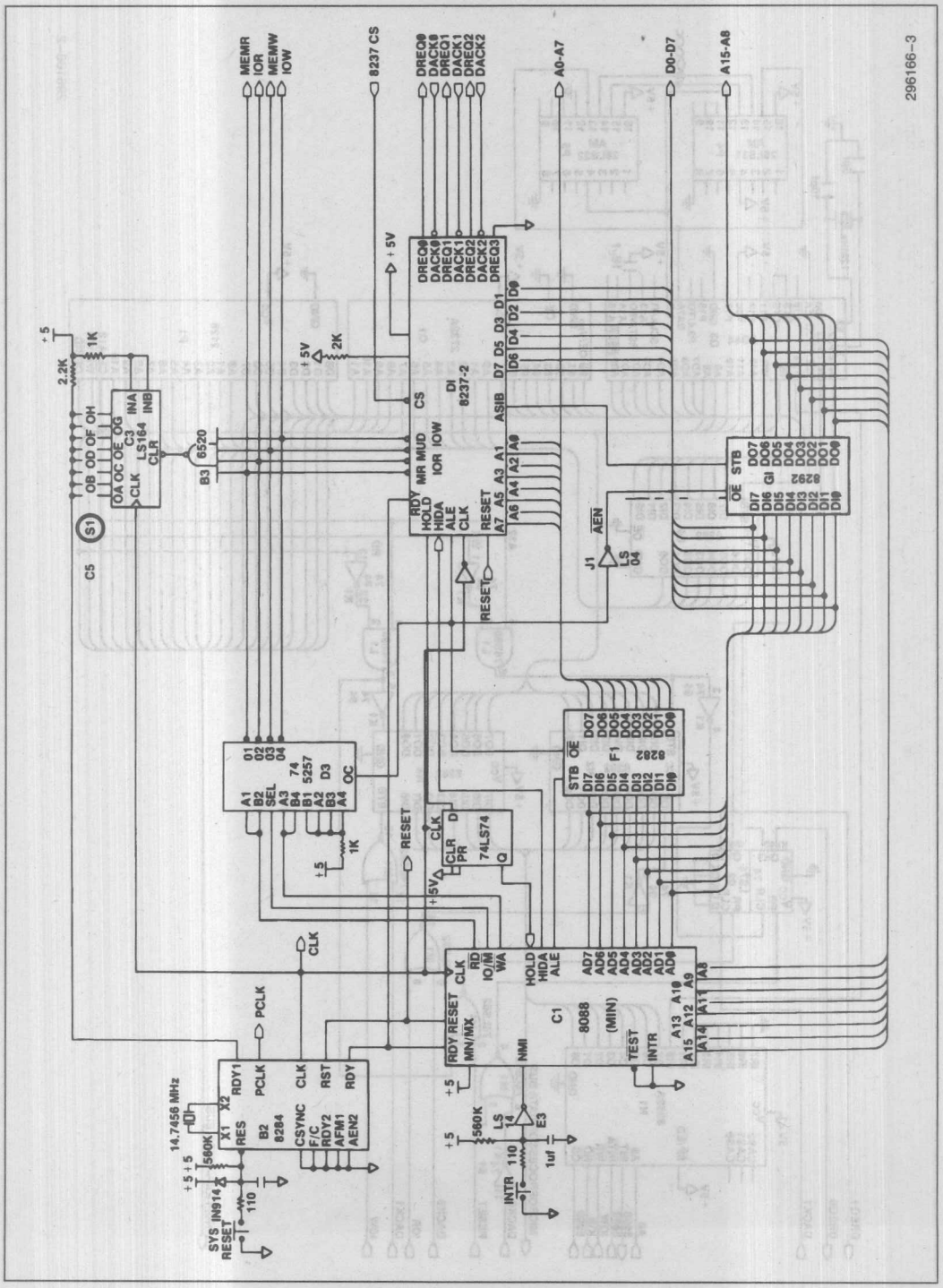


Figure 3. 8088 Min Mode System

Table 1. Transmit and Receive Software for an 8044/8088 System

```
LOC  OBJ    LINE     SOURCE
                1  $debug    title    (8044/8088 INTERFACE)
                2
                3
  0000          4  FIRST_BYTE      BIT     0       ; FLAG
                5
  0000          6            ORG    0
  0000 8024     7            SJMP   INIT
                8
  0026          9            ORG    26H
               10
  0026 7581AA  11  INIT:      MOV    SP, #170        ; INITIALIZE STACK
  0029 75B800  12             MOV    IP, #00         ; ALL INTERRUPTS ARE EQUAL PRIORITY
  002C 75C954  13             MOV    SMD, #54H       ; TIMER 1 OVERFLOW, NRZI, PRE-FRAME SYNC
  002F 758844  14             MOV    TCON, #44H      ; EDGE TRIGGERED EXTERNAL INTERRUPT 1
               15                                     ; LEVEL TRIGGERED EXTERNAL INTERRUPT 0
               16                                     ; TIMER 1 ON
  0032 758DEC  17             MOV    TH1, #0ECH      ; INITIALIZE TIMER, 3125 BPS
  0035 758920  18             MOV    TMOD, #20H      ; TIMER 1 AUTO RELOAD
               19
  0038 75DC6A  20             MOV    TBS, #106       ; SET UP SIU PARAMETER REGISTERS
  003B 75DB40  21             MOV    TBL, #64
  003E 75CC2A  22             MOV    RBS, #42
  0041 75CB40  23             MOV    RBL, #64
  0044 75CE55  24             MOV    STAD, #55H
  0047 75DA11  25             MOV    TCB, #00010001B ; RR, P/F=1
               26
  004A 901000  27             MOV    DPTR, #1000H    ; DPTR POINTS TO TRI-STATE LATCH
  004D D200    28             SETB   FIRST_BYTE      ; FLAG TO INDICATE FIRST BYTE
               29                                     ; FOR RECEIVE INTERRUPT ROUTINE
  004F D2CE    30             SETB   RBE             ; READY TO RECEIVE
  0051 75A894  31             MOV    IE, #10010100B  ; ENABLE RECEIVE DMA AND SIU INTERRUPT
               32
  0054 80FE    33             SJMP   $               ; WAIT HERE FOR INTERRUPTS
               34
  0056 80FE    35  ERROR:     SJMP   ERROR
               36 +1 $EJ
               37  ;*************************      SUBROUTINES      *********************************
               38
  0058 85CD29  39  SEND:      MOV    41, RFL         ; FIRST BYTE IN BLOCK IS COUNT
  005B 7929    40             MOV    R1, #41         ; POINT TO BLOCK OF DATA
  005D AACD    41             MOV    R2, RFL         ; LOAD COUNT
  005F 0A      42             INC    R2
  0060 D2A8    43             SETB   EX0             ; ENABLE DMA TRANSMIT INTERRUPT
  0062 22      44             RET
               45
               46
               47
               48  ;*********************    INTERRUPT SERVICE ROUTINES    **********************
               49
  0063         50  LOC_TMP    SET    $               ; SET UP INTERRUPT TABLE JUMP
  0013         51             ORG    0013H
  0013 020063  52             LJMP   RECEIVE_DMA
  0063         53             ORG    LOC_TMP
               54
               55  RECEIVE_DMA:
```

296166-69

Table 1. Transmit and Receive Software for an 8044/8088 System (Continued)

```
                    56
0063 10000E         57              JBC     FIRST_BYTE, L1 ; THE FIRST BYTE TRANSFERRED IS THE COUNT
                    58
0066 E0             59              MOVX    A, @DPTR       ; READ THE LATCH
0067 F6             60              MOV     @R0, A         ; PUT IT IN TRANSMIT BUFFER
0068 08             61              INC     R0
0069 DA08           62              DJNZ    R2, L2         ; AFTER READING BYTES,
                    63
006B D2CF           64              SETB    TBF            ; SEND DATA
006D D2CD           65              SETB    RTS
006F D200           66              SETB    FIRST_BYTE
0071 C2AA           67              CLR     EX1
                    68
0073 32             69      L2:     RETI
                    70
0074 786A           71      L1:     MOV     R0, #106       ; R0 IS A POINTER TO THE TRANSMIT
                    72                                     ; BUFFER STARTING ADDRESS
0076 E0             73              MOVX    A, @DPTR       ; PUT THE FIRST BYTE INTO
0077 FA             74              MOV     R2, A          ; R2 FOR THE COUNT
0078 32             75              RETI
                    76
  0079              77      LOC_TMP SET     $
0003                78              ORG     0003H
0003 020079         79              LJMP    TRANSMIT_DMA
0079                80              ORG     LOC_TMP
                    81
                    82      TRANSMIT_DMA
                    83
0079 E7             84              MOV     A, @R1         ; READ BYTE OUT OF THE RECEIVE BUFFER
007A F0             85              MOVX    @DPTR, A       ; WRITE IT TO THE LATCH
007B 09             86              INC     R1
007C DA08           87              DJNZ    R2, L3         ; WHEN ALL BYTES HAVE BEEN SENT
                    88
007E C2A8           89              CLR     IE. 0          ; DISABLE INTERRUPT
0080 C294           90              CLR     P1. 4          ; CAUSE 8088 INTERRUPT TO TERMINATE DMA
0082 D294           91              SETB    P1. 4
0084 D2CE           92              SETB    RBE            ; ENABLE RECEIVER AGAIN
                    93
0086 32             94      L3:     RETI
                    95
                    96
                    97
  0087              98      LOC_TMP SET     $
0023                99              ORG     0023H
0023 020087        100              LJMP    SERIAL_INT
0087               101              ORG     LOC_TMP
                   102
                   103      SERIAL_INT:
                   104
0087 30CE06        105              JNB     RBE, RCV       ; WAS A FRAME RECEIVED
008A 30CF0B        106              JNB     TBF, XMIT      ; WAS A FRAME TRANSMITTED
008D 020056        107              LJMP    ERROR          ; IF NEITHER ERROR
                   108
0090 20CBC3        109      RCV:    JB      BOV, ERROR     ; IF BUFFER OVERRUN THEN ERROR
0093 1158          110              CALL    SEND           ; SEND THE FRAME TO THE 8088
0095 C2CC          111              CLR     SI
0097 32            112              RETI
                   113
0098 C2CC          114      XMIT:   CLR     SI
```

296166-70

```
009A D2AA      115          SETB   EX1
009C 32        116          RETI
               117
               118          END
```

SYMBOL TABLE LISTING

NAME	TYPE	VALUE	ATTRIBUTES
BOV	B ADDR	00C8H.3	A
ERROR	C ADDR	0056H	A
EX0	B ADDR	00A8H.0	A
EX1	B ADDR	00A8H.2	A
FIRST_BYTE	B ADDR	0020H.0	A
IE	D ADDR	00A8H	A
INIT	C ADDR	0026H	A
IP	D ADDR	00B8H	A
L1	C ADDR	0074H	A
L2	C ADDR	0073H	A
L3	C ADDR	0086H	A
LOC_TMP	C ADDR	0087H	A
P1	D ADDR	0090H	A
RBE	B ADDR	00C8H.6	A
RBL	D ADDR	00CBH	A
RBS	D ADDR	00CCH	A
RCV	C ADDR	0090H	A
RECEIVE_DMA	C ADDR	0063H	A
RFL	D ADDR	00CDH	A
RTS	B ADDR	00C8H.5	A
SEND	C ADDR	0058H	A
SERIAL_INT	C ADDR	0087H	A
SI	B ADDR	00C8H.4	A
SMD	D ADDR	00C9H	A
SP	D ADDR	0081H	A
STAD	D ADDR	00CEH	A
TBF	B ADDR	00C8H.7	A
TBL	D ADDR	00DBH	A
TBS	D ADDR	00DCH	A
TCB	D ADDR	00DAH	A
TCON	D ADDR	0088H	A
TH1	D ADDR	008DH	A
TMOD	D ADDR	0089H	A
TRANSMIT_DMA	C ADDR	0079H	A
XMIT	C ADDR	0098H	A

REGISTER BANK(S) USED: 0, TARGET MACHINE(S): 8044

ASSEMBLY COMPLETE, NO ERRORS FOUND

296166-71

Table 2. PL/M-86 Compiler RUPI/8088 Interface Example

```
SERIES-III PL/M-86 V1.0 COMPILATION OF MODULE RUPI_88
OBJECT MODULE PLACED IN :F1:R88.OBJ
COMPILER INVOKED BY:  PLM86.86 :F1:R88.SRC


              $DEBUG
              $TITLE  ('RUPI/8088 INTERFACE EXAMPLE')

  1           RUPI_88: DO;
  2   1           DECLARE

                  LIT                 LITERALLY 'LITERALLY',
                  TRUE                LIT       '01H',
                  FALSE               LIT       '00H',

                  RECV_BUFFER(64)     BYTE,
                  XMIT_BUFFER(64)     BYTE,
                  I                   BYTE,
                  WAIT                BYTE,

                                /* 8237 PORTS*/

                  MASTER_CLEAR_37     LIT       'OFFDDH',
                  COMMAND_37          LIT       'OFFD8H',
                  ALL_MASK_37         LIT       'OFFDFH',
                  SINGLE_MASK_37      LIT       'OFFDAH',
                  STATUS_37           LIT       'OFFD8H',
                  REQUEST_REG_37      LIT       'OFFD9H',
                  MODE_REG_37         LIT       'OFFDBH',
                  CLEAR_BYTE_PTR_37   LIT       'OFFDCH',

                  CH0_ADDR            LIT       'OFFD0H',
                  CH0_COUNT           LIT       'OFFD1H',
                  CH1_ADDR            LIT       'OFFD2H',
                  CH1_COUNT           LIT       'OFFD3H',
                  CH2_ADDR            LIT       'OFFD4H',
                  CH2_COUNT           LIT       'OFFD5H',
                  CH3_ADDR            LIT       'OFFD6H',
                  CH3_COUNT           LIT       'OFFD7H',

                          /* 8237 BIT ASSIGNMENTS */

                  CH0_SEL             LIT       '00H',
                  CH1_SEL             LIT       '01H',
                  CH2_SEL             LIT       '02H',
                  CH3_SEL             LIT       '03H',
                  WRITE_XFER          LIT       '04H',
                  READ_XFER           LIT       '08H',
                  DEMAND_MODE         LIT       '00H',
                  SINGLE_MODE         LIT       '40H',
                  BLOCK_MODE          LIT       '80H',
                  SET_MASK            LIT       '04H',

              $EJECT
                                /* 8259 PORTS */

                  STATUS_POLL_59      LIT       'OFFE0H',
                  ICW1_59             LIT       'OFFE0H',
                  OCW1_59             LIT       'OFFE1H',
                  OCW2_59             LIT       'OFFE0H',
                  OCW3_59             LIT       'OFFE0H',
                  ICW2_59             LIT       'OFFE1H',
                  ICW3_59             LIT       'OFFE1H',
                  ICW4_59             LIT       'OFFE1H';

                          /* INTERRUPT SERVICE ROUTINE */

  3   1       OFF_RECV_DMA:   PROCEDURE   INTERRUPT 32;

  4   2           OUTPUT(SINGLE_MASK_37)=40H;
  5   2           WAIT=FALSE;
  6   2           END;
```

296166-4

Table 2. PL/M-86 Compiler RUPI/8088 Interface Example (Continued)

```
      7   1           DISABLE;

                              /* INITIALIZE 8237 */

      8   1           OUTPUT(MASTER_CLEAR_37)          =0;
      9   1           OUTPUT(COMMAND_37)               =040H;
     10   1           OUTPUT(ALL_MASK_37)              =0FH;
     11   1           OUTPUT(MODE_REG_37)              =(SINGLE_MODE OR WRITE_XFER OR CH0_SEL);
     12   1           OUTPUT(MODE_REG_37)              =(SINGLE_MODE OR READ_XFER OR CH1_SEL);
     13   1           OUTPUT(CLEAR_BYTE_PTR_37)        =0;
     14   1           OUTPUT(CH0_ADDR)                 =00H;
     15   1           OUTPUT(CH0_ADDR)                 =40H;
     16   1           OUTPUT(CH0_COUNT)                =64;
     17   1           OUTPUT(CH0_COUNT)                =00;
     18   1           OUTPUT(CH1_ADDR)                 =40H;
     19   1           OUTPUT(CH1_ADDR)                 =40H;
     20   1           OUTPUT(CH1_COUNT)                =64;
     21   1           OUTPUT(CH1_COUNT)                =00;

                              /* INITIALIZE 8259 */

     22   1           OUTPUT(ICW1_59)                  =13H; /*SINGLE MODE, EDGE TRIGGERED
                                                              INPUT, 8086 INTERRUPT TYPE*/
     23   1           OUTPUT(ICW2_59)                  =20H; /*INTERRUPT TYPE 32*/
     24   1           OUTPUT(ICW4_59)                  =03H; /*AUTO-EOI*/
     25   1           OUTPUT(OCW1_59)                  =0FEH; /*ENABLE INTERRUPT LEVEL 0*/

                $EJECT
     26   1           CALL SET$INTERRUPT  (32,OFF_RECV_DMA); /*LOAD INTERRUPT VECTOR LOCATION*/

     27   1           XMIT_BUFFER(0)=64; /*THE FIRST BYTE IN THE BLOCK OF DATA IS THE NUMBER
                                           OF BYTES TO BE TRANSFERED; NOT INCLUDING THE FIRST BYTE*/

     28   1           DO  I= 1 TO 64;  /* FILL UP THE XMIT_BUFFER WITH DATA */
     29   2           XMIT_BUFFER(I)=I;
     30   2           END;

     31   1           OUTPUT(ALL_MASK_37)=0FCH;   /*ENABLE CHANNEL 1 AND 2 */

     32   1           ENABLE;

     33   1           WAIT=TRUE;
     34   1           DO WHILE WAIT;
     35   2           END;                    /* A BLOCK OF DATA WILL BE TRANSFERRED TO THE RUPI.
                                                 WHEN THE RUPI RECEIVES A BLOCK OF DATA IT WILL
                                                 SEND IT TO THE 8088 MEMORY AND INTERRUPT THE 8088.
                                                 THE INTERRUPT SERVICE ROUTINE WILL SHUT OFF THE DMA
                                                 CONTROLLER AND SET 'WAIT' FALSE */

     36   1           DO WHILE 1;
     37   2           END;

     38   1       END;


MODULE INFORMATION:

     CODE AREA SIZE      = 00D7H    215D
     CONSTANT AREA SIZE  = 0000H      0D
     VARIABLE AREA SIZE  = 0082H    130D
     MAXIMUM STACK SIZE  = 001EH     30D
     124 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
```

296166-5

A HIGH PERFORMANCE NETWORK USING THE 8044

2.0 INTRODUCTION

This section describes the design of an SDLC data link using the 8044 (RUPI) to implement a primary station and a secondary station. The design was implemented and tested. The following discussion assumes that the reader understands the 8044 and SDLC. This section is divided into two parts. First the data link design example is discussed. Second the software modules used to implement the data link are described. To help the reader understand the discussion of the software, flow charts and software listings are displayed in Appendix A and Appendix B, respectively.

APPLICATION DESCRIPTION

This particular data link design example uses a two wire half-duplex multidrop topology as shown in Figure 4. In an SDLC multidrop topology the primary station communicates with each secondary station. The secondary stations communicate only to the primary. Because of this hierarchial architecture, the logical topology for an SDLC multidrop is a star as shown in Figure 5. Although the physical topology of this data link is multidrop, the easiest way to understand the information flow is to think of the logical (star) topology. The term data link in this case refers to the logical communication pathways between the primary station and the secondary stations. The data links are shown in Figure 5 as two way arrows.

The application example uses dumb async terminals to interface to the SDLC network. Each secondary station has an async terminal connected to it. The secondary stations are in effect protocol converters which allows any async terminal to communicate with any other async terminal on the network. The secondary stations use an 8044 with a UART to convert SDLC to async. Figure 6 displays a block diagram of the data link. The primary station, controls the data link. In addition to data link control the primary provides a higher level layer which is a path control function or networking layer. The primary serves as a message exchange or switch. It receives information from one secondary station and retransmits it to another secondary station. Thus a virtual end to end connection is made between any two secondary stations on the network.

Three separate software modules were written for this network. The first module is a Secondary Station Driver (SSD) which provides an SDLC data link interface and a user interface. This module is a general purpose driver which requires application software to run it. The user interface to the driver provides four functions: OPEN, CLOSE, TRANSMIT, and SIU_RECV. Using these four functions properly will allow any application software to communicate over this SDLC data link without knowing the details of SDLC. The secondary station driver uses the 8044's AUTO mode.

The second module is an example of application software which is linked to the secondary station driver. This module drives the 8215A, buffers data, and interfaces with the secondary station driver's user interface.

The third module is a primary station, which is a standalone program (i.e., it is not linked to any other module). The primary station uses the 8044's NON-AUTO or FLEXIBLE mode. In addition to controlling the data link it acts as a message switch. Each time a secondary station transmits a frame, it places the destination address of the frame in the first byte of the information or I field. When the primary station receives a frame, it removes the first byte in the I field and retransmits the frame to the secondary station whose address matches this byte.

This network provides two complete layers of the OSI (Open Systems Interconnection) reference model: the physical layer and the data link layer. The physical layer implementation uses the RS-422 electrical interface. The mechanical medium consists of ribbon cable and connectors. The data link layer is defined by SDLC. SDLC's use of acknowledgements and frame numbering guarantees that messages will be received in the same order in which they were sent. It also guarantees message integrity over the data link. However this network will not guarantee secondary to secondary message delivery, since there are acknowledgements between secondary stations.

2.1 Hardware

The schematic of the hardware is given in Figure 7. The 8251A is used as an async communications controller, in support of the 8044. TxRDY and RxRDY on the 8251A are both tied to the two available external interrupts of the 8044 since the secondary station driver is totally interrupt driven. The 8044 buffers the data and some variables in a 2016 (2K x 8 static RAM). The 8254 programmable interval timer is employed as a programmable baud rate generator and system clock driver for the 8251A. The third output from the 8254 could be used as an external baud rate generator for the 8044. The 2732A shown in the diagram was not used

since the software for both the primary and secondary stations used far less than the 4K bytes provided on the 8744. For the async interface, the standard RS-232 mechanical and electrical interface was used. For the SDLC channel, a standard two wire three state RS-422 driver is used. A DIP switch connected to one of the available ports on the 8044 allows the baud rate, parity, and stop bits to be changed on the async interface. The primary station hardware does not use the USART, 8254, nor the RS-232 drivers.

2.2 SDLC Basic Repertoire

The SDLC commands and responses implemented in the data link include the SDLC Basic Repertoire as defined in the IBM SDLC General Information manual. Table 3 shows the commands and responses that the primary and the secondary station in this data link design recognize and send.

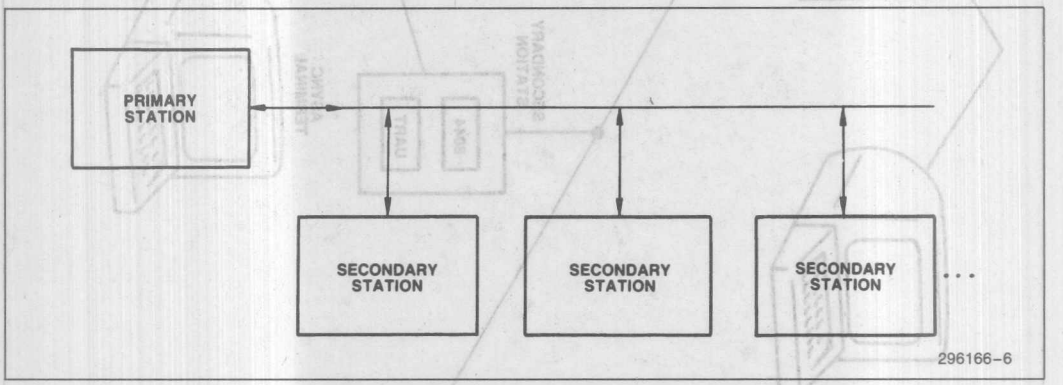


Figure 4. SDLC Multidrop Topology

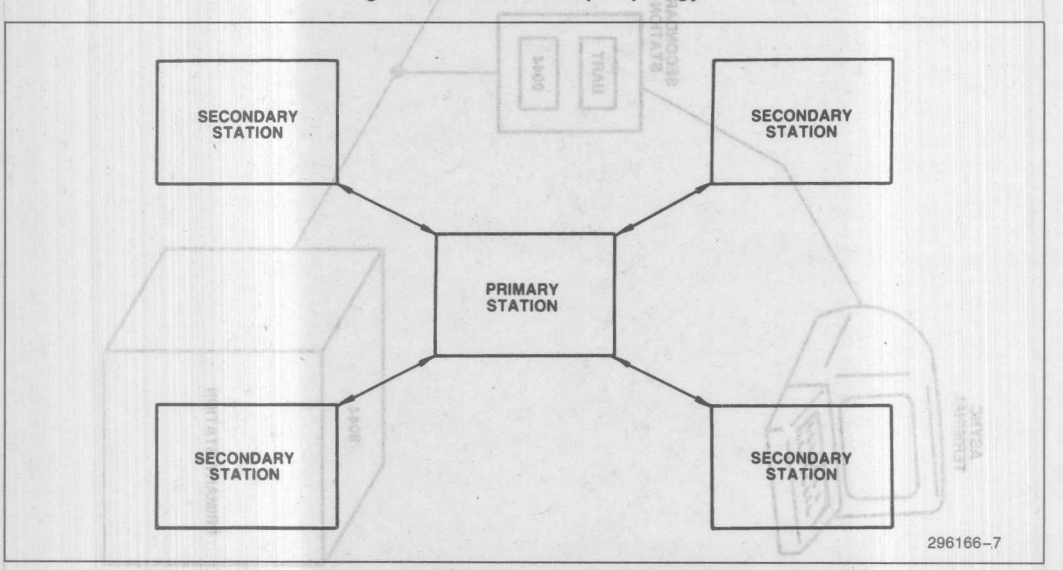


Figure 5. SDLC Logical Topology

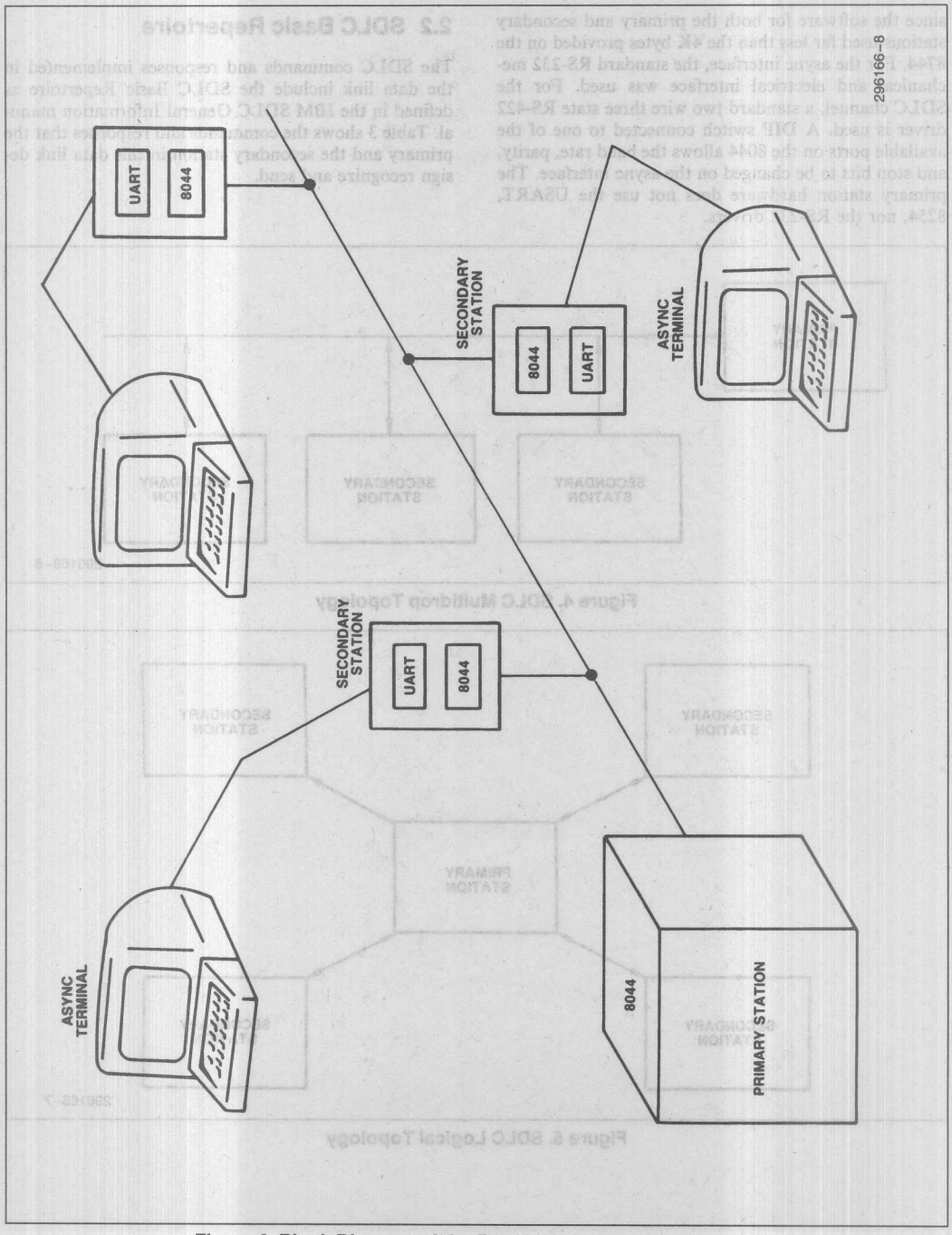


Figure 6. Block Diagram of the Data Link Application Example

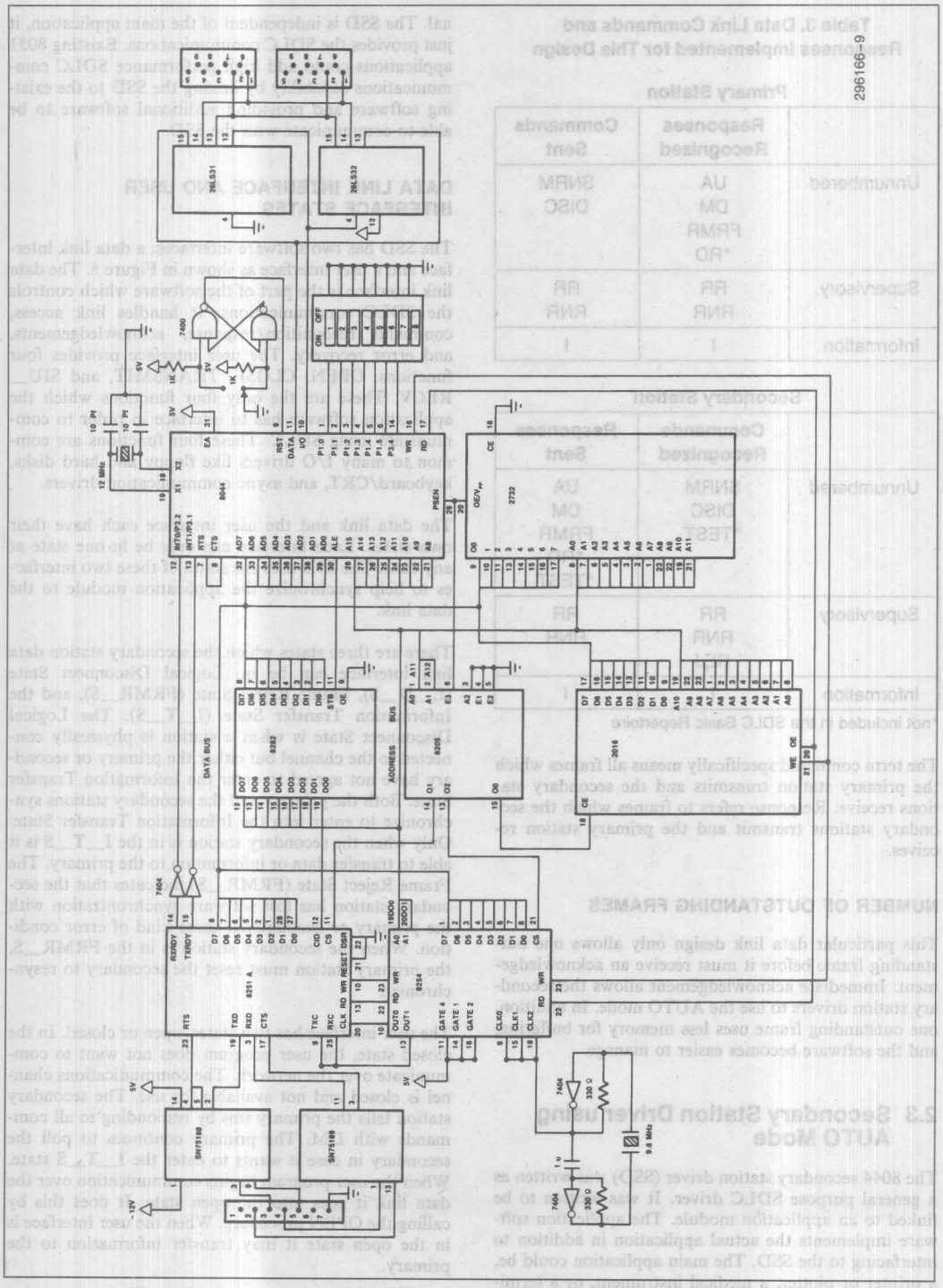


Figure 7. Schematic of Async/SDLC Secondary Station Protocol Converter

Responses Implemented for This Design

Primary Station

	Responses Recognized	Commands Sent
Unnumbered	UA DM FRMR *RD	SNRM DISC
Supervisory	RR RNR	RR RNR
Information	I	I

Secondary Station

	Commands Recognized	Responses Sent
Unnumbered	SNRM DISC *TEST	UA DM FRMR *RD *TEST
Supervisory	RR RNR REJ	RR RNR
Information	I	I

*not included in the SDLC Basic Repertoire

The term command specifically means all frames which the primary station transmits and the secondary stations receive. Response refers to frames which the secondary stations transmit and the primary station receives.

NUMBER OF OUTSTANDING FRAMES

This particular data link design only allows one outstanding frame before it must receive an acknowledgement. Immediate acknowledgement allows the secondary station drivers to use the AUTO mode. In addition, one outstanding frame uses less memory for buffering, and the software becomes easier to manage.

2.3 Secondary Station Driver using AUTO Mode

The 8044 secondary station driver (SSD) was written as a general purpose SDLC driver. It was written to be linked to an application module. The application software implements the actual application in addition to interfacing to the SSD. The main application could be, a printer or plotter, a medical instrument, or a terminal just provides the SDLC communications. Existing 8051 applications could add high performance SDLC communications capability by linking the SSD to the existing software and providing additional software to be able to communicate with the SSD.

DATA LINK INTERFACE AND USER INTERFACE STATES

The SSD has two software interfaces: a data link interface and a user interface as shown in Figure 8. The data link interface is the part of the software which controls the SDLC communications. It handles link access, command recognition/response, acknowledgements, and error recovery. The user interface provides four functions: OPEN, CLOSE, TRANSMIT, and SIU_RECV. These are the only four functions which the application software has to interface in order to communicate using SDLC. These four functions are common to many I/O drivers like floppy and hard disks, keyboard/CRT, and async communication drivers.

The data link and the user interface each have their own states. Each interface can only be in one state at any time. The SSD uses the states of these two interfaces to help synchronize the application module to the data link.

There are three states which the secondary station data link interface can be in: Logical Disconnect State (L_D_S), Frame Reject State (FRMR_S), and the Information Transfer State (I_T_S). The Logical Disconnect State is when a station is physically connected to the channel but either the primary or secondary have not agreed to enter the Information Transfer State. Both the primary and the secondary stations synchronize to enter into the Information Transfer State. Only when the secondary station is in the I_T_S is it able to transfer data or information to the primary. The Frame Reject State (FRMR_S) indicates that the secondary station has lost software synchronization with the primary or encountered some kind of error condition. When the secondary station is in the FRMR_S, the primary station must reset the secondary to resynchronize.

The user interface has two states, open or closed. In the closed state, the user program does not want to communicate over the network. The communications channel is closed and not available for use. The secondary station tells the primary this by responding to all commands with DM. The primary continues to poll the secondary in case it wants to enter the I_T_S state. When the user program begins communication over the data link it goes into the open state. It does this by calling the OPEN procedure. When the user interface is in the open state it may transfer information to the primary.

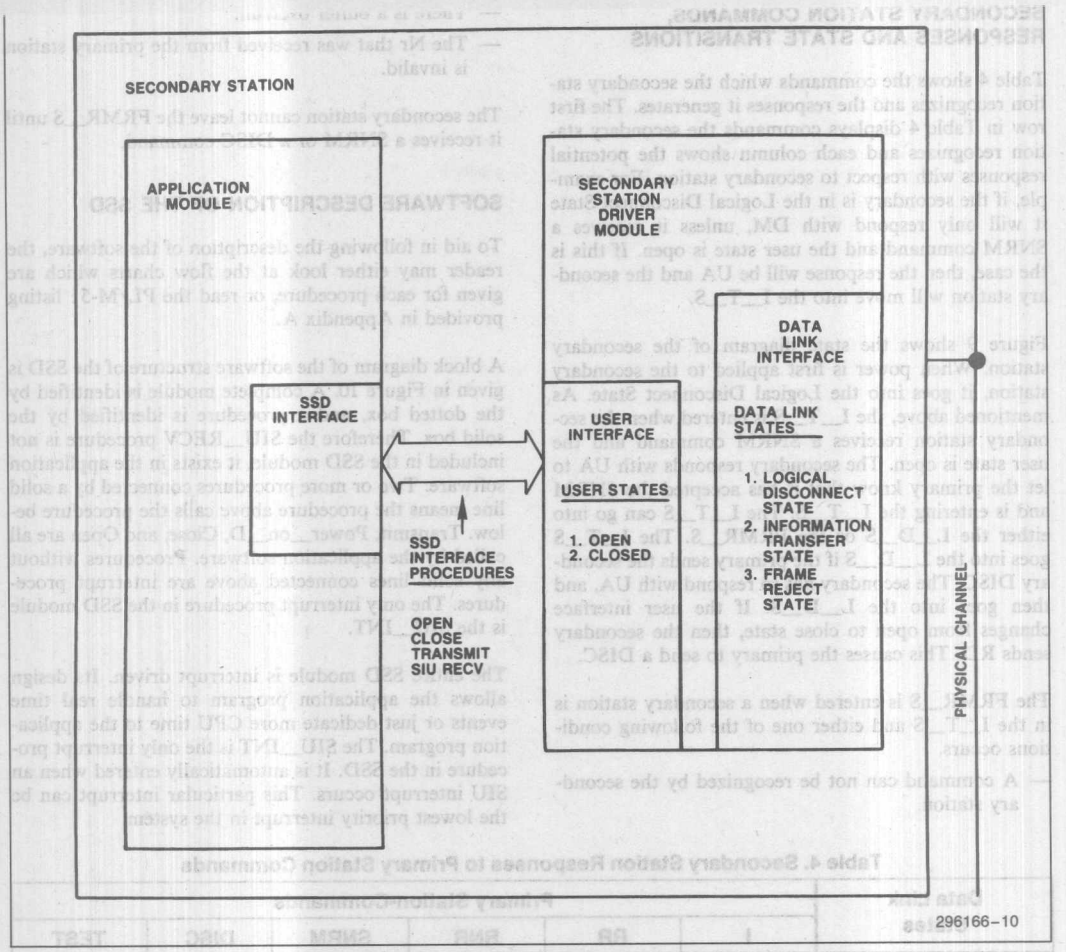


Figure 8. Secondary Station Software Modules

SECONDARY STATION COMMANDS, RESPONSES AND STATE TRANSITIONS

Table 4 shows the commands which the secondary station recognizes and the responses it generates. The first row in Table 4 displays commands the secondary station recognizes and each column shows the potential responses with respect to secondary station. For example, if the secondary is in the Logical Disconnect State it will only respond with DM, unless it receives a SNRM command and the user state is open. If this is the case, then the response will be UA and the secondary station will move into the I__T__S.

Figure 9 shows the state diagram of the secondary station. When power is first applied to the secondary station, it goes into the Logical Disconnect State. As mentioned above, the I__T__S is entered when the secondary station receives a SNRM command and the user state is open. The secondary responds with UA to let the primary know that it has accepted the SNRM and is entering the I__T__S. The I__T__S can go into either the L__D__S or the FRMR__S. The I__T__S goes into the L__D__S if the primary sends the secondary DISC. The secondary has to respond with UA, and then goes into the L__D__S. If the user interface changes from open to close state, then the secondary sends RD. This causes the primary to send a DISC.

The FRMR__S is entered when a secondary station is in the I__T__S and either one of the following conditions occurs.

— A command can not be recognized by the secondary station.

— There is a buffer overrun.

— The Nr that was received from the primary station is invalid.

The secondary station cannot leave the FRMR__S until it receives a SNRM or a DISC command.

SOFTWARE DESCRIPTION OF THE SSD

To aid in following the description of the software, the reader may either look at the flow charts which are given for each procedure, or read the PL/M-51 listing provided in Appendix A.

A block diagram of the software structure of the SSD is given in Figure 10. A complete module is identified by the dotted box, and a procedure is identified by the solid box. Therefore the SIU__RECV procedure is not included in the SSD module, it exists in the application software. Two or more procedures connected by a solid line means the procedure above calls the procedure below. Transmit, Power__on__D, Close, and Open are all called by the application software. Procedures without any solid lines connected above are interrupt procedures. The only interrupt procedure in the SSD module is the SIU__INT.

The entire SSD module is interrupt driven. Its design allows the application program to handle real time events or just dedicate more CPU time to the application program. The SIU__INT is the only interrupt procedure in the SSD. It is automatically entered when an SIU interrupt occurs. This particular interrupt can be the lowest priority interrupt in the system.

Table 4. Secondary Station Responses to Primary Station Commands

Data Link States	Primary Station-Commands					
	I	RR	RNR	SNRM	DISC	TEST
Information Transfer State	I RR RNR RD FRMR	I RR RNR RD FRMR	I RR RNR RD FRMR	 RD UA	 UA	 RD Test
Logical Disconnect State	DM	DM	DM	DM UA	DM	DM
Frame Reject State	FRMR	FRMR	FRMR	 UA	 UA	FRMR

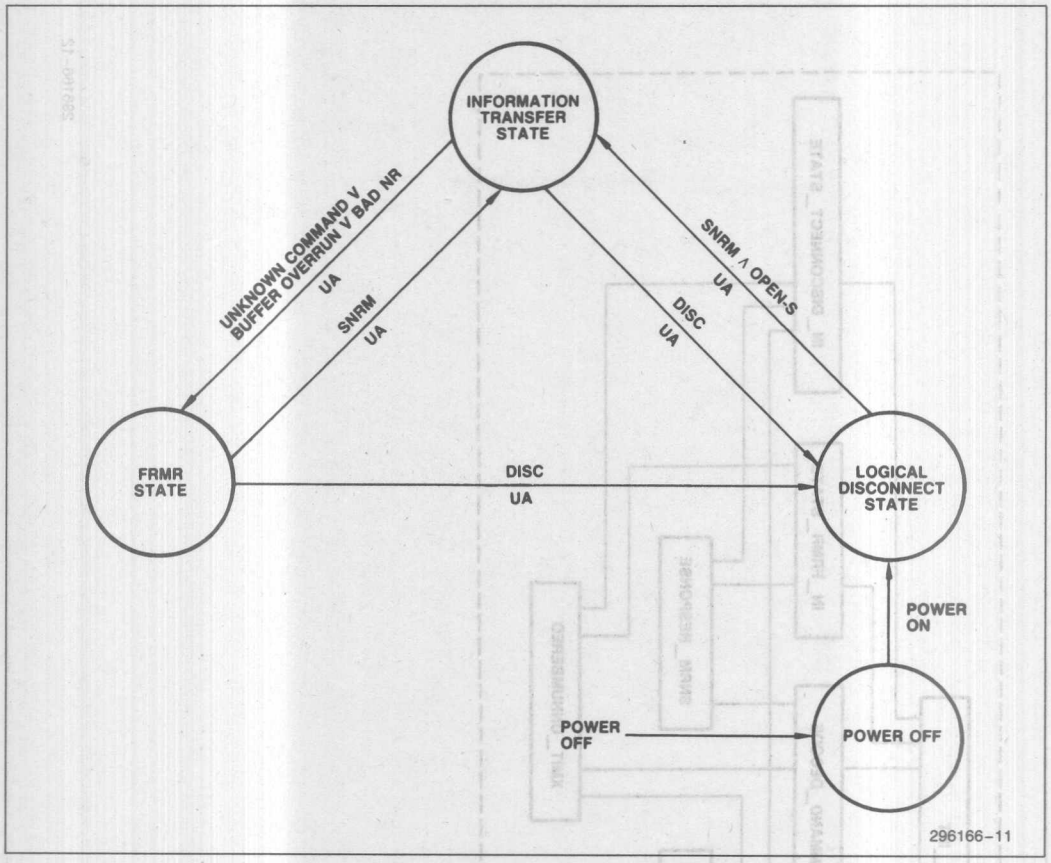


Figure 9. State Diagram of Secondary Station

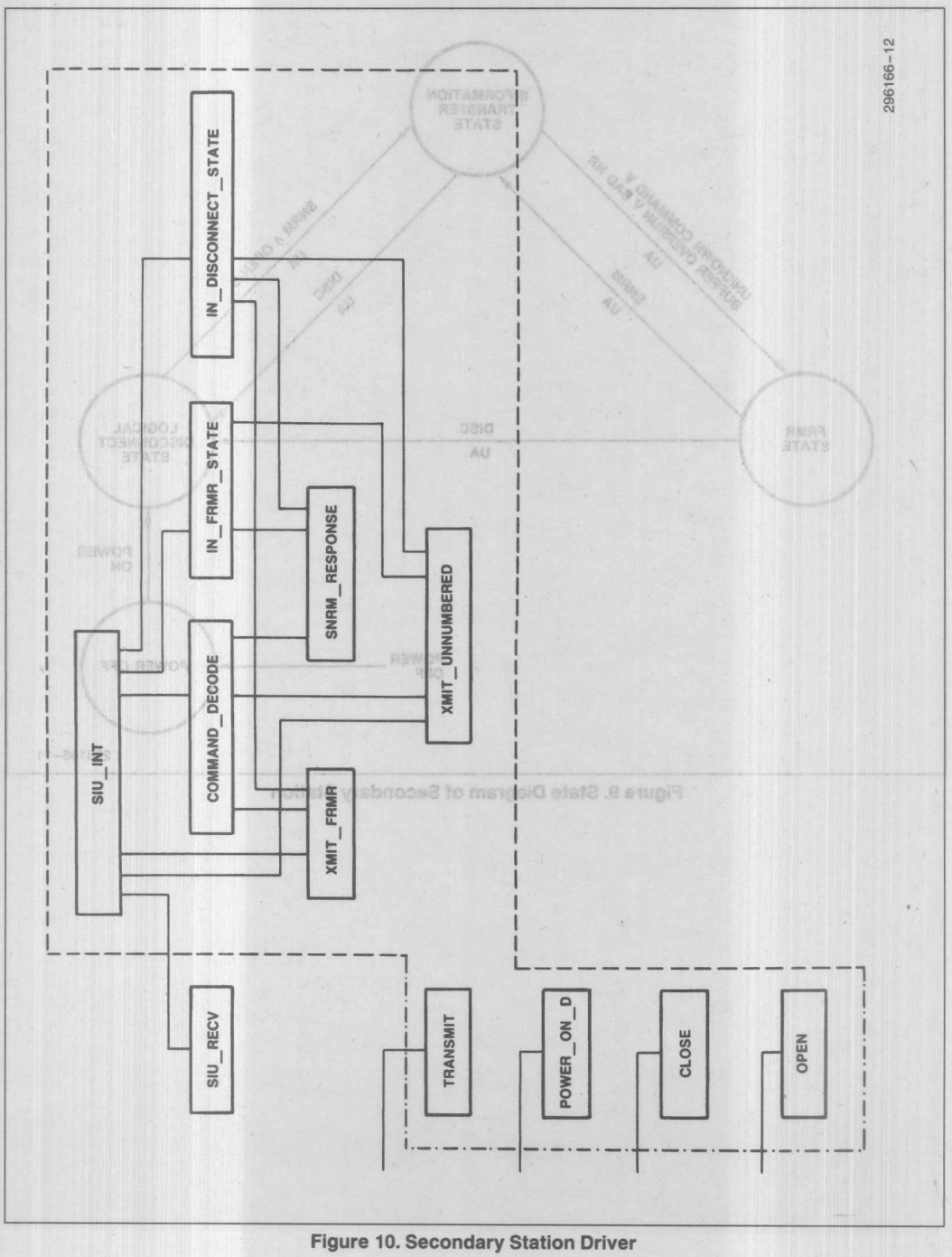


Figure 10. Secondary Station Driver

SSD INITIALIZATION

Upon reset the application software is entered first. The application software initializes its own variables then calls Power_On_D which is the SSD's initialization routine. The SSD's initialization sets up the transmit and receive data buffer pointers (TBS and RBS), the receive buffer length (RBL), and loads the State variables. The STATION_STATE begins in the L_D_S state, and the USER_STATE begins in the closed state. Finally Power_On_D initializes XMIT_BUFFER_EMPTY which is a bit flag. This flag serves as a semaphore between the SSD and the application software to indicate the status of the on chip transmit buffer. The SSD does not set the station address. It is the application software's responsibility to do this. After initialization, the SSD is read to respond to all of the primary station commands. Each time a frame is received with a matching station address and a good CRC, the SIU_INT procedure is entered.

SIU_INT PROCEDURE

The first thing the SIU_INT procedure clears is the serial interrupt_bit (SI) in the STS register. If the SIU_INT procedure returns with this bit set, another SI interrupt will occur.

The SIU_INT procedure is branches three independent cases. The first case is entered if the STATION_STATE is not in the I_T_S. If this is true, then the SIU is not in the AUTO mode, and the CPU will have to respond to the primary on its own. (Remember that the AUTO mode is entered when the STATION_STATE enters into I_T_S.) If the STATION_STATE is in the I_T_S, then either the SIU has just left the AUTO mode, or is still in the AUTO mode. This is the second and third case, respectively.

In the first case, if the STATION_STATE is not in the I_T_S, then it must be in either the L_D_S or the FRMR_S. In either case a separate procedure is called based on which state the station is in. The In_Disconnect_State procedure sends to the primary a DM response, unless it received a SNRM command and the USER_STATE equals open. In that case the SIU sends a UA and enters into the I_T_S. The In_FRMR_State procedure will send the primary the FRMR response unless it received either a DISC or an SNRM. If the primary's command was a DISC, then the secondary will send a UA and enter into the L_D_S. If the primary's command was a SNRM, then the secondary will send a UA, enter into the I_T_S, and clear NSNR register.

For the second case, if the STATION_STATE is in the I_T_S but the SIU left the AUTO mode, then the CPU must determine why the AUTO mode was exited, and generate a response to the primary. There are four reasons for the SIU to automatically leave the AUTO mode. The following is a list of these reasons, and the responses given by the SSD based on each reason.

1. The SIU has received a command field it does not recognize.

   Response: If the CPU recognizes the command, it generates the appropriate response. If neither the SIU nor the CPU recognize the command, then a FRMR response is sent.

2. The SIU has received a Sequence Error Sent (SES = 1 in NSNR register). $Nr(P) \neq Ns(S) + 1$, and $Nr(P) \neq Ns(S)$.

   Response: Send FRMR.

3. A buffer overrun has occurred. BOV = 1 in STS register.

   Response: Send FRMR.

4. An I frame with data was received while RPB = 1.

   Response: Go back into AUTO mode and send an AUTO mode response

In addition to the above reasons, there is one condition where the CPU forces the SIU out of the AUTO mode. This is discussed in the SSD's User Interface Procedures section in the CLOSED procedure description

Finally, case three is when the STATION_STATE is in the I_T_S and the AUTO mode. The CPU first looks at the TBF bit. If this bit is 0 then the interrupt may have been caused by a frame which was transmitted and acknowledged. Therefore the XMIT_BUFFER_EMPTY flag is set again, indicating that the application software can transmit another frame.

The other reason this section of code could be entered is if a valid I frame was received. When a good I frame is received the RBE bit equals 0. This means that the receiver is disabled. If the primary were to poll the 8044 while RBE = 0, it would time out since no response would be given. Time outs reduce network throughput. To improve network performance, the CPU first sets RBP, then sets RBE. Now when the primary polls the 8044 an immediate RNR response is given. At this point the SSD calls the application software procedure SIU_RECV and passes the length of the data as a parameter. The SIU_RECV procedure reads the data out of the receive buffer then returns to the SSD module. Now that the receive information has been transferred, RBP can be cleared.

COMMAND_DECODE PROCEDURE

The Command_Decode procedure is called from the SIU_INT procedure when the STATION_STATE = I_T_S and the SIU left the AUTO mode as a result of not being able to recognize the receive control byte. Commands which the SIU AUTO mode does not

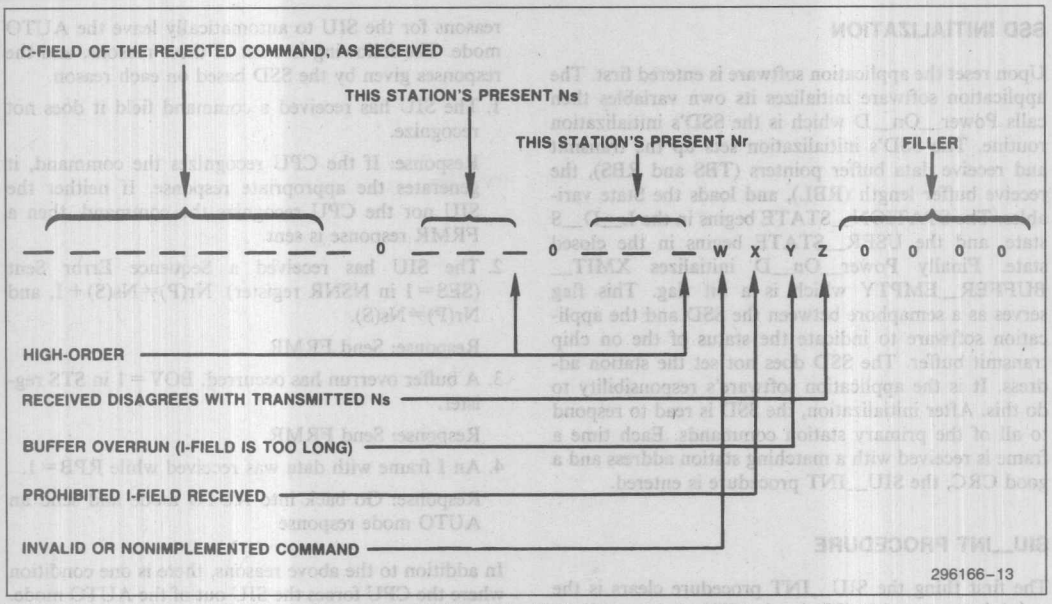


Figure 11. Information Field of the FRMR Response, as Transmitted

recognize are handled here. The commands recognized in this procedure are: SNRM, DISC, and TEST. Any other command received will generate a Frame Reject with the nonimplemented command bit set in the third data byte of the FRMR frame. Any additional unnumbered frame commands which the secondary station is going to implement, should be implemented in this procedure.

IF an SNRM is received the command__decode procedure calls the SNRM__Response procedure. The SNRM__Response procedure sets the STATION__STATE = I__T__S, clears the NSNR register and responds with a UA frame. If a DISC is received, the command__decode procedure sets the STATION__STATE = L__D__S, and responds with a UA frame. When a TEST frame is received, and there is no buffer overrun, the command__decode procedure responds with a TEST frame retransmitting the same data it received. However if a TEST frame is received and there is a buffer overrun, then a TEST frame will be sent without any data, instead of a FRMR with the buffer overrun bit set.

FRAME REJECT PROCEDURES

There are two procedures which handle the FRMR state: XMIT__FRMR and IN__FRMR__STATE. XMIT__FRMR is entered when the secondary station first goes into the FRMR state. The frame reject response frame contains the FRMR response in the command field plus three additional data bytes in the I field. Figure 11 displays the format for the three data bytes in the I field of a FRMR response. The XMIT__FRMR procedure sets up the Frame Reject response frame based on the parameter REASON which is passed to it. Each place in the SSD code that calls the XMIT__FRMR procedure, passes the REASON that this procedure was called, which in turn is communicated to the primary station. The XMIT__FRMR procedure uses three bytes of internal RAM which it initializes for the correct response. The TBS and TBL registers are then changed to point to the FRMR buffer so that when a response is sent these three bytes will be included in the I field.

The IN__FRMR__STATE procedure is called by the SIU__INT procedure when the STATION__STATE already is in the FRMR state and a response is required. The IN__FRMR__STATE procedure will only allow two commands to remove the secondary station from the FRMR state: SNRM and DISC. Any other command which is received while in the FRMR state will result in a FRMR response frame.

XMIT__UNNUMBERED PROCEDURE

This is a general purpose transmit procedure, used only in the FLEXIBLE mode, which sends unnumbered responses to the primary. It accepts the control byte as a parameter, and also expects the TBL register to be set before the procedure is called. This procedure waits until the frame has been transmitted before returning. If

this procedure returned before the transmit interrupt was generated, the SIU_INT routine would be entered. The SIU_INT routine would not be able to distinguish this condition.

SSD's User Interface Procedures—OPEN, CLOSE, TRANSMIT, SIU_RECV—are discussed in the following section.

The OPEN procedure is the simplest of all, it changes the USER_STATE to OPEN_S then returns. This lets the SSD know that the user wants to open the channel for communications. When the SSD receives a SNRM command, it checks the USER_STATE. If the USER_STATE is open, then the SSD will respond with a UA, and the STATION_STATE enters the I_T_S.

The CLOSE procedure is also simple, it changes the USER_STATE to CLOSED_S and sets the AM bit to 0. Note that when the CPU sets the AM bit to 0 it puts the SIU out of the AUTO mode. This event is asynchronous to the events on the network. As a result an I frame can be lost. This is what can happen.

1. AM is set to 0 by the CLOSE Procedure.
2. An I frame is received and an SI interrupt occurs.
3. The SIU_INT procedure enters case 2 (STATION_STATE = I_T_S, and AM = 0).
4. Case 2 detects that the USER_STATE = CLOSED_S, sends an RD response and ignores the fact that an I frame was received.

Therefore it is advised to never call the CLOSE procedure or take the SIU out of the AUTO mode when it is receiving I frames or an I frame will be lost.

For both the TRANSMIT and SIU_RECV procedures, it is the application software's job to put data into the transmit buffer, and take data out of the receive buffer. The SSD does not transfer data in or out of its transmit or receive buffers because it does not know what kind of buffering the application software is implementing. What the SSD does do is notify the application software when the transmit buffer is empty, XMIT_BUFFER_EMPTY = 1, and when the receive buffer is full.

One of the functions that the SSD performs to synchronize the application software to the SDLC data link. However some of the synchronization must also be done by the application software. Remember that the SSD does not want to hang up the application software waiting for some event to occur on the SDLC data link, therefore the SSD always returns to the application software as soon as possible.

For example, when the application software calls the OPEN procedure, the SSD returns immediately. The application software thinks that the SDLC channel is now open and it can transmit. This is not the case. For the channel to be open, the SSD must receive an SNRM from the primary and respond with a UA. However, the SSD does not want to hang up the application software waiting for an SNRM from the primary before returning from the OPEN procedure. When the TRANSMIT procedure is called, the SSD expects the STATION_STATE to be in the I_T_S. If it isn't, the SSD refuses to transmit the data. The TRANSMIT procedure first checks to see if the USER_STATE is open. If not, the USER_STATE_CLOSED parameter is passed back to the application module. The next thing TRANSMIT checks is the STATION_STATE. If this is not open, then TRANSMIT passes back LINK_DISCONNECTED. This means that the USER_STATE is open, but the SSD hasn't received an SNRM command from the primary yet. Therefore, the application software should wait awhile and try again. Based on network performance, one knows the maximum amount of time it will take for a station to be polled. If the application software waits this length of time and tries again but still gets a LINK_DISCONNECTED parameter passed back, higher level recovery must be implemented.

Before loading the transmit buffer and calling the TRANSMIT procedure, the application software must check to see that XMIT_BUFFER_EMPTY = 1. This flag tells the application software that it can write new data into the transmit buffer and call the TRANSMIT procedure. After the application software has verified that XMIT_BUFFER_EMPTY = 1, it fills the transmit buffer with the data and calls the TRANSMIT procedure passing the length of the buffer as a parameter. The TRANSMIT procedure checks for three reasons why it might not be able to transmit the frame. If any of these three reasons are true, the TRANSMIT procedure returns a parameter explaining why it couldn't send the frame. If the application software receives one of these responses, it must rectify the problem and try again. Assuming these three conditions are false, then the SSD clears XMIT_BUFFER_EMPTY, attempts to send the data and returns the parameter DATA_TRANSMITTED. XMIT_BUFFER_EMPTY will not be set to 1 again until the data has been transmitted and acknowledged.

The SIU_RECV procedure must be incorporated into the application software module. When a valid I frame is received by the SIU, it calls the SIU_RECV procedure and passes the length of the received data as a parameter. The SIU_RECV procedure must remove all of the data from the receive buffer before returning to the SIU_INT procedure.

LINKING UP TO THE SSD

Figure 12 shows the necessary parts to include in a PL/M-51 application program that will be linked to the SSD module. RL51 is used to link and locate the SSD and application modules. The command line used to do this is:

```
RL51 SSD.obj,filename.obj,PLM51.LIB TO
filename & RAMSIZE(192)

$registerbank(0)
user$mod: do;
$include (reg44.dcl)
declare
 lit                literally 'literally',
 buffer_length      lit       '60',
 siu_xmit_buffer
 (buffer_length)    byte    external idata,
 siu_recv_buffer
 (buffer_length)    byte      external,
 xmit_buffer_empty bit        external;

/* external procedures */

power_on_d: procedure  external;
end power_on_d;

close: procedure       external using 1;
end close;

open: procedure        external using 1;
end open;

transmit: procedure
(xmit_buffer_length) byte    external;
 declare xmit_buffer_length   byte;
end transmit;

/* local procedures */

siu_recv: procedure (length)    using 1;
public
 declare   length byte,
           •
           •
           •
end siu_recv;
```

Figure 12. Applications Module Link Information

PL/M-51 AND REGISTER BANKS

The 8044 has four register banks. PL/M-51 assumes that an interrupt procedure never uses the same bank as the procedure it interrupts. The USING attribute of a procedure, or the $REGISTERBANK control, can be used to ensure that.

The SSD module uses the $REGISTERBANK(1) attribute. Some procedures are modified with the USING attribute based on the register bank level of the calling procedure.

2.4 Application Module; ASYNC to SDLC Protocol Converter

One of the purposes of this application module is to demonstrate how to interface software to the SSD. Another purpose is to implement and test a practical application. This application software performs I/O with an async terminal through a USART, buffers data, and also performs I/O with the SSD. In addition, it allows the user on the async terminal to: set the station address, set the destination address, and go online and offline. Setting the station address sets the byte in the STAD register. The destination address is the first byte in the I field. Going online or offline results in either calling the OPEN or CLOSE procedure respectively.

After the secondary station powers up, it enters the 'terminal mode', which accepts data from the terminal. However, before any data is sent, the user must configure the station. The station address and destination address must be set, and the station must be placed online. To configure the station the ESC character is entered at the terminal which puts the protocol converter into the 'configure mode'. Figure 13 shows the menu which appears on the terminal screen.

```
(/)8044 Secondary Station
  /

1 - Set the Station Address
2 - Set the Destination Address
3 - Go Online
4 - Go Offline
5 - Return to terminal mode
    Enter option __
```

Figure 13. Menu for the Protocol Converter

In the terminal mode data is buffered up in the secondary station. A Line Feed character 'LF' tells the secondary station to send an I frame. If more than 60 bytes are buffered in the secondary station when a 'LF' is received, the applications software packetizes the data into 60 bytes or less per frame. If a LF is entered when the station is offline, an error message comes on the screen which says 'Unable to Get Online'.

The secondary station also does error checking on the async interface for Parity, Framing Error, and Overrun Error. If one of these errors are detected, an error message is displayed on the terminal screen.

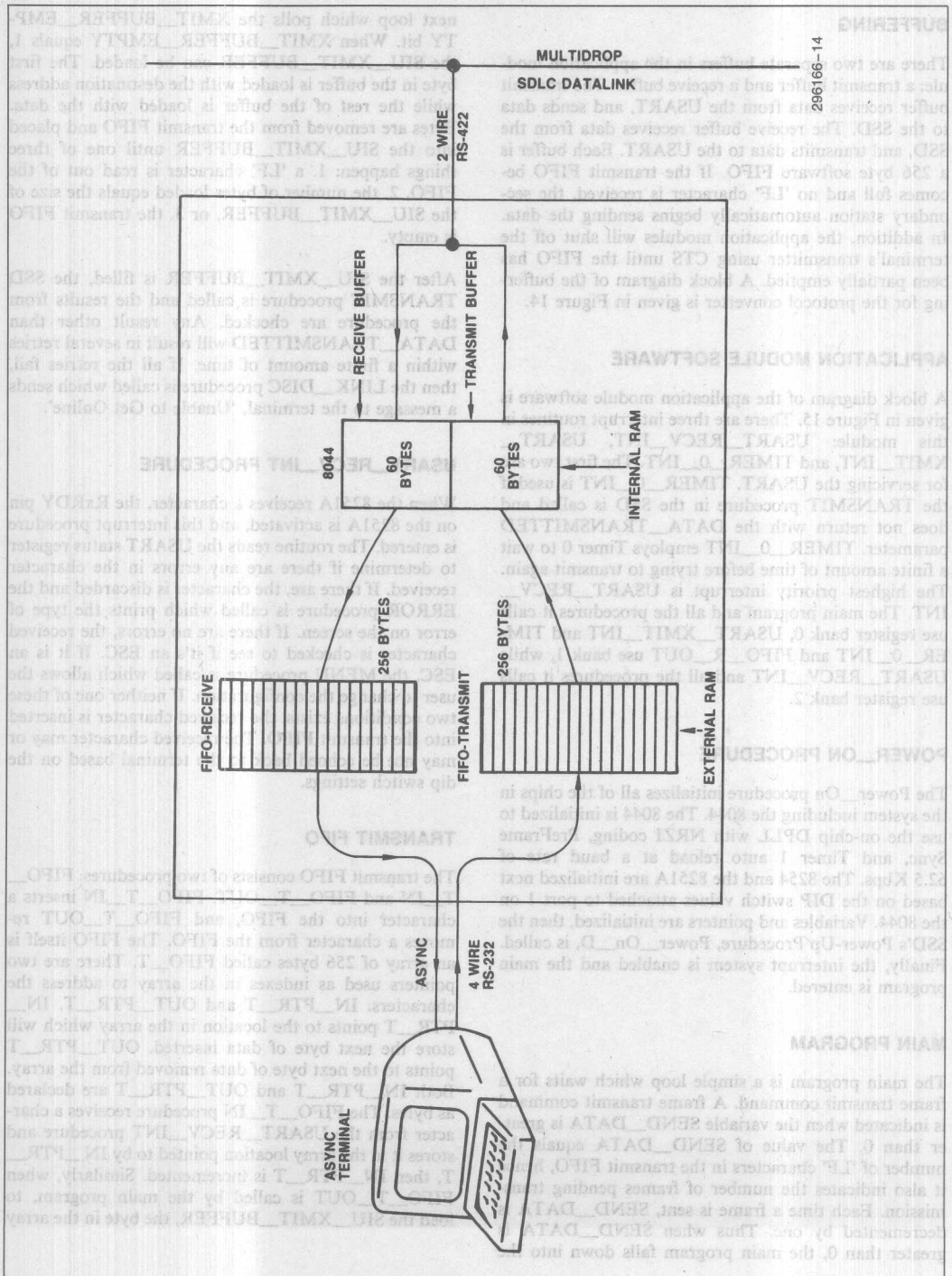


Figure 14. Block Diagram of Secondary Station Protocol Converter Illustrating Buffering

BUFFERING

There are two separate buffers in the application module: a transmit buffer and a receive buffer. The transmit buffer receives data from the USART, and sends data to the SSD. The receive buffer receives data from the SSD, and transmits data to the USART. Each buffer is a 256 byte software FIFO. If the transmit FIFO becomes full and no 'LF' character is received, the secondary station automatically begins sending the data. In addition, the application modules will shut off the terminal's transmitter using CTS until the FIFO has been partially emptied. A block diagram of the buffering for the protocol converter is given in Figure 14.

APPLICATION MODULE SOFTWARE

A block diagram of the application module software is given in Figure 15. There are three interrupt routines in this module: USART_RECV_INT, USART_XMIT_INT, and TIMER_0_INT. The first two are for servicing the USART. TIMER_0_INT is used if the TRANSMIT procedure in the SSD is called and does not return with the DATA_TRANSMITTED parameter. TIMER_0_INT employs Timer 0 to wait a finite amount of time before trying to transmit again. The highest priority interrupt is USART_RECV_INT. The main program and all the procedures it calls use register bank 0, USART_XMIT_INT and TIMER_0_INT and FIFO_R_OUT use bank 1, while USART_RECV_INT and all the procedures it calls use register bank 2.

POWER_ON PROCEDURE

The Power_On procedure initializes all of the chips in the system including the 8044. The 8044 is initialized to use the on-chip DPLL with NRZI coding, PreFrame Sync, and Timer 1 auto reload at a baud rate of 62.5 Kbps. The 8254 and the 8251A are initialized next based on the DIP switch values attached to port 1 on the 8044. Variables and pointers are initialized, then the SSD's Power-Up Procedure, Power_On_D, is called. Finally, the interrupt system is enabled and the main program is entered.

MAIN PROGRAM

The main program is a simple loop which waits for a frame transmit command. A frame transmit command is indicated when the variable SEND_DATA is greater than 0. The value of SEND_DATA equals the number of 'LF' characters in the transmit FIFO, hence it also indicates the number of frames pending transmission. Each time a frame is sent, SEND_DATA is decremented by one. Thus when SEND_DATA is greater than 0, the main program falls down into the next loop which polls the XMIT_BUFFER_EMPTY bit. When XMIT_BUFFER_EMPTY equals 1, the SIU_XMIT_BUFFER can be loaded. The first byte in the buffer is loaded with the destination address while the rest of the buffer is loaded with the data. Bytes are removed from the transmit FIFO and placed into the SIU_XMIT_BUFFER until one of three things happen: 1. a 'LF' character is read out of the FIFO, 2. the number of bytes loaded equals the size of the SIU_XMIT_BUFFER, or 3. the transmit FIFO is empty.

After the SIU_XMIT_BUFFER is filled, the SSD TRANSMIT procedure is called and the results from the procedure are checked. Any result other than DATA_TRANSMITTED will result in several retries within a finite amount of time. If all the retries fail, then the LINK_DISC procedure is called which sends a message to the terminal, 'Unable to Get Online'.

USART_RECV_INT PROCEDURE

When the 8251A receives a character, the RxRDY pin on the 8251A is activated, and this interrupt procedure is entered. The routine reads the USART status register to determine if there are any errors in the character received. If there are, the character is discarded and the ERROR procedure is called which prints the type of error on the screen. If there are no errors, the received character is checked to see if it's an ESC. If it is an ESC, the MENU procedure is called which allows the user to change the configuration. If neither one of these two conditions exists, the received character is inserted into the transmit FIFO. The received character may or may not be echoed back to the terminal based on the dip switch settings.

TRANSMIT FIFO

The transmit FIFO consists of two procedures: FIFO_T_IN and FIFO_T_OUT. FIFO_T_IN inserts a character into the FIFO, and FIFO_T_OUT removes a character from the FIFO. The FIFO itself is an array of 256 bytes called FIFO_T. There are two pointers used as indexes in the array to address the characters: IN_PTR_T and OUT_PTR_T. IN_PTR_T points to the location in the array which will store the next byte of data inserted. OUT_PTR_T points to the next byte of data removed from the array. Both IN_PTR_T and OUT_PTR_T are declared as bytes. The FIFO_T_IN procedure receives a character from the USART_RECV_INT procedure and stores it in the array location pointed to by IN_PTR_T, then IN_PTR_T is incremented. Similarly, when FIFO_T_OUT is called by the main program, to load the SIU_XMIT_BUFFER, the byte in the array

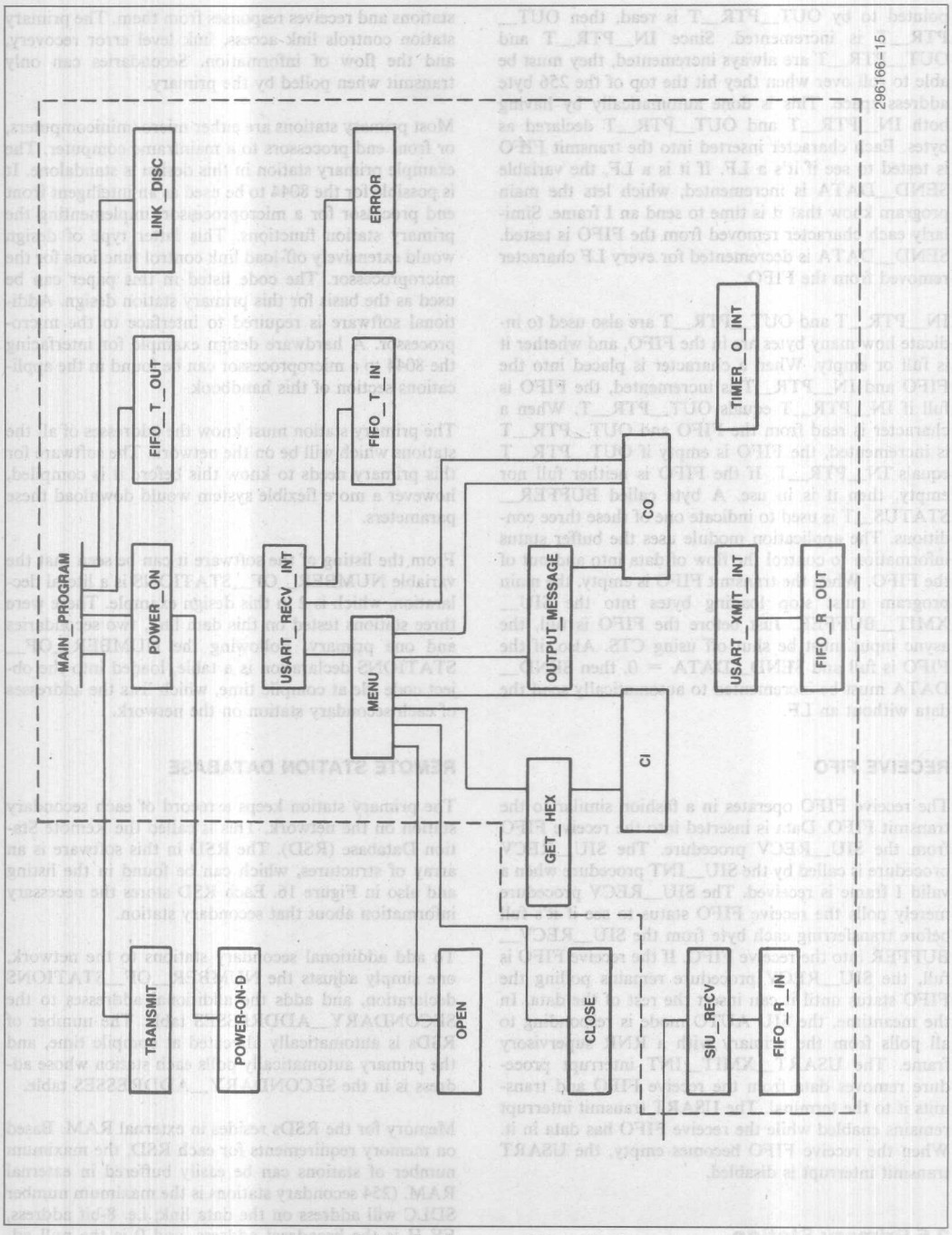


Figure 15. Block Diagram of User Software

pointed to by OUT_PTR_T is read, then OUT_PTR_T is incremented. Since IN_PTR_T and OUT_PTR_T are always incremented, they must be able to roll over when they hit the top of the 256 byte address space. This is done automatically by having both IN_PTR_T and OUT_PTR_T declared as bytes. Each character inserted into the transmit FIFO is tested to see if it's a LF. If it is a LF, the variable SEND_DATA is incremented, which lets the main program know that it is time to send an I frame. Similarly each character removed from the FIFO is tested. SEND_DATA is decremented for every LF character removed from the FIFO.

IN_PTR_T and OUT_PTR_T are also used to indicate how many bytes are in the FIFO, and whether it is full or empty. When a character is placed into the FIFO and IN_PTR_T is incremented, the FIFO is full if IN_PTR_T equals OUT_PTR_T. When a character is read from the FIFO and OUT_PTR_T is incremented, the FIFO is empty if OUT_PTR_T equals IN_PTR_T. If the FIFO is neither full nor empty, then it is in use. A byte called BUFFER_STATUS_T is used to indicate one of these three conditions. The application module uses the buffer status information to control the flow of data into and out of the FIFO. When the transmit FIFO is empty, the main program must stop loading bytes into the SIU_XMIT_BUFFER. Just before the FIFO is full, the async input must be shut off using CTS. Also, if the FIFO is full and SEND_DATA = 0, then SEND_DATA must be incremented to automatically send the data without an LF.

RECEIVE FIFO

The receive FIFO operates in a fashion similar to the transmit FIFO. Data is inserted into the receive FIFO from the SIU_RECV procedure. The SIU_RECV procedure is called by the SIU_INT procedure when a valid I frame is received. The SIU_RECV procedure merely polls the receive FIFO status to see if it's full before transferring each byte from the SIU_RECV_BUFFER into the receive FIFO. If the receive FIFO is full, the SIU_RECV procedure remains polling the FIFO status until it can insert the rest of the data. In the meantime, the SIU AUTO mode is responding to all polls from the primary with a RNR supervisory frame. The USART_XMIT_INT interrupt procedure removes data from the receive FIFO and transmits it to the terminal. The USART transmit interrupt remains enabled while the receive FIFO has data in it. When the receive FIFO becomes empty, the USART transmit interrupt is disabled.

2.5 Primary Station

The primary station is responsible for controlling the data link. It issues commands to the secondary stations and receives responses from them. The primary station controls link access, link level error recovery, and the flow of information. Secondaries can only transmit when polled by the primary.

Most primary stations are either micro/minicomputers, or front end processors to a mainframe computer. The example primary station in this design is standalone. It is possible for the 8044 to be used as an intelligent front end processor for a microprocessor, implementing the primary station functions. This latter type of design would extensively off-load link control functions for the microprocessor. The code listed in this paper can be used as the basis for this primary station design. Additional software is required to interface to the microprocessor. A hardware design example for interfacing the 8044 to a microprocessor can be found in the applications section of this handbook.

The primary station must know the addresses of all the stations which will be on the network. The software for this primary needs to know this before it is compiled, however a more flexible system would download these parameters.

From the listing of the software it can be seen that the variable NUMBER_OF_STATIONS is a literal declaration, which is 2 in this design example. There were three stations tested on this data link, two secondaries and one primary. Following the NUMBER_OF_STATIONS declaration is a table, loaded into the object code file at compile time, which lists the addresses of each secondary station on the network.

REMOTE STATION DATABASE

The primary station keeps a record of each secondary station on the network. This is called the Remote Station Database (RSD). The RSD in this software is an array of structures, which can be found in the listing and also in Figure 16. Each RSD stores the necessary information about that secondary station.

To add additional secondary stations to the network, one simply adjusts the NUMBER_OF_STATIONS declaration, and adds the additional addresses to the SECONDARY_ADDRESSES table. The number of RSDs is automatically allocated at compile time, and the primary automatically polls each station whose address is in the SECONDARY_ADDRESSES table.

Memory for the RSDs resides in external RAM. Based on memory requirements for each RSD, the maximum number of stations can be easily buffered in external RAM. (254 secondary stations is the maximum number SDLC will address on the data link; i.e. 8-bit address, FF H is the broadcast address, and 0 is the null address. Each RSD uses 70 bytes of RAM. 70 x 254 = 17,780.)

The station state, in the RSD structure, maintains the status of the secondary. If this byte indicates that the secondary is in the DISCONNECT_S, then the primary tries to put the station in the I_T_S by sending an SNRM. If the response is a UA then the station state changes into the I_T_S. Any other frame received results in the station state remaining in the DISCONNECT_S. When the RSD indicates that the station state is in the I_T_S, the primary will send either an I, RR, or RNR command, depending on the local and remote buffer status. When the station state equals GO_TO_DISC the primary will send a DISC command. If the response is a UA frame, the station state will change to DISCONNECT_S, else the station state will remain in GO_TO_DISC. The station state is set to GO_TO_DISC when one of the following responses occur:

1. A receive buffer overrun in the primary.
2. An I frame is received and Nr(P) $\neq$ Ns(S).
3. An I frame or a Supervisory frame is received and Ns(P) + 1 $\neq$ Nr(S) and Ns(P) $\neq$ Nr(S).
4. A FRMR response is received.
5. An RD response is received.
6. An unknown response is received.

The send count (Ns) and receive count (Nr) are also maintained in the RSD. Each time an I frame is sent by the primary and acknowledged by the secondary, Ns is incremented. Nr is incremented each time a valid I frame is received. BUFFER_STATUS indicates the status of the secondary station's buffer. If an RR response is received, BUFFER_STATUS is set to BUFFER_READY. If a RNR response is received, BUFFER_STATUS is set to BUFFER_NOT_READY.

BUFFERING

The buffering for the primary station is as follows: within each RSD is a 64 byte array buffer which is initially empty. When the primary receives an I frame, it looks for a match between the first byte of the I frame and the addresses of the secondaries on the network. If a match exists, the primary places the data in the RSD buffer of the destination station. The INFO_LENGTH in the RSD indicates how many bytes are in the buffer. If INFO_LENGTH equals 0, then the buffer is empty. The primary can buffer only one I frame per station. If a second I frame is received while the addressed secondary's RSD buffer is full, the primary cannot receive any more I frames. At this point the primary continues to poll the secondaries using RNR supervisory frame.

PRIMARY STATION SOFTWARE

A block diagram of the primary station software is shown in Figure 17. The primary station software consists of a main program, one interrupt routine, and several procedures. The POWER_ON procedure begins by initializing the SIU's DMA and enabling the receiver. Then each RSD is initialized. The DPLL and the timers are set, and finally the TIMER 0 interrupt is enabled.

The main program consists of an iterative do loop within a do forever loop. The iterative do loop polls each secondary station once through the do loop. The variable STATION_NUMBER is the counter for the iterative do statement which is also used as an index to the array of RSD structures. The primary station issues one command and receives one response from every secondary station each time through the loop. The first statement in the loop loads the secondary station address, indexed by STATION_NUMBER into the array of the RSD structures. Now when the primary sends a command, it will have the secondary's address in the address field of the frame. The automatic address recognition feature is used by the primary to recognize the response from the secondary.

Next, the main program determines the secondary station's state. Based on this state, the primary knows what command to send. If the station is in the DISCONNECT_S, the primary calls the SNRM_P procedure to try and put the secondary in the I_T_S. If the station state is in the GO_TO_DISC state, the DISC_P is called to try and put the secondary in the L_D_S. If the secondary is in neither one of the above two states, then it is in the I_T_S. When the secondary is in the I_T_S, the primary could send one of three commands: I, RR, or RNR. If the RSD's buffer has data in it, indicated by INFO_LENGTH being greater than zero, and the secondary's BUFFER_STATUS equals BUFFER_READY, then an I frame will be sent. Else if RPB = 0, an RR supervisory frame will be sent. If neither one of these cases is true, then an RNR will be sent. The last statement in the main program checks the RPB bit. If set to one, the BUFFER_TRANSFER procedure is called, which transfers the data from the SIU receive buffer to the appropriate RSD buffer.

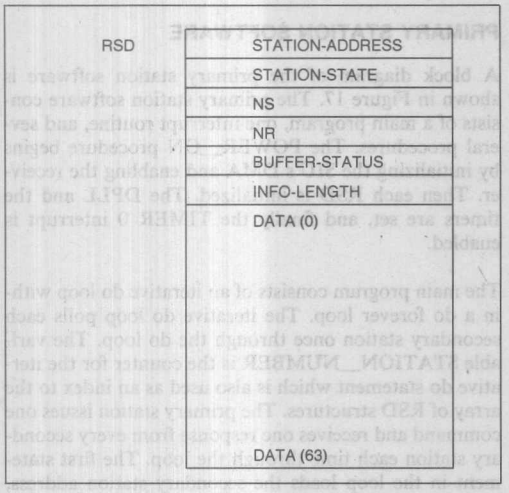


Figure 16. Remote Station Database Structure

RECEIVE TIME OUT

Each time a frame is transmitted, the primary sets a receive time out timer; Timer 0. If a response is not received within a certain time, the primary returns to the main program and continues polling the rest of the stations. The minimum length of time the primary should wait for a response can be calculated as the sum of the following parameters.

1. Propagation time to the secondary station
2. Clear-to-send at the secondary station's DCE
3. Appropriate time for secondary station processing
4. Propagation time from the secondary station
5. Maximum frame length time

The clear-to-send time and the propagation time are negligible for a local network at low bit rates. However, the turnaround time and the maximum frame length time are significant factors. Using the 8044 secondaries in the AUTO mode minimizes turnaround time. The maximum frame length time comes from the fact the 8044 does not generate an interrupt from a received frame until it has been completely received, and the CRC is verified as correct. This means that the time-out is bit rate dependent.

Ns AND Nr CHECK PROCEDURES

Each time an I frame or supervisory frame is received, the Nr field in the control byte must be checked. Since this data link only allows one outstanding frame, a valid Nr would satisfy either one of two equations; Ns(P) + 1 = Nr(S) the I frame previously sent by the primary is acknowledged, Ns(P) = Nr(S) the I frame previously sent is not acknowledged. If either one of these two cases is true, the CHECK_NR procedure returns a parameter of TRUE; otherwise a FALSE parameter is returned. If an acknowledgement is received, the Ns byte in the RSD structure is incremented, and the Information buffer may be cleared. Otherwise the information buffer remains full.

When an I frame is received, the Ns field has to be checked also. If Nr(P) = Ns(S), then the procedure returns TRUE, otherwise a FALSE is returned.

RECEIVE PROCEDURE

The receive procedure is called when a supervisory or information frame is sent, and a response is received before the time-out period. The RECEIVE procedure can be broken down into three parts. The first part is entered if an I frame is received. When an I frame is received, Ns, Nr and buffer overrun are checked. If there is a buffer overrun, or there is an error in either Ns or Nr, then the station state is set to GO_TO_DISC. Otherwise Nr in the RSD is incremented, the receive field length is saved, and the RPB bit is set. By incrementing the Nr field, the I frame just received is acknowledged the next time the primary polls the secondary with an I frame or a supervisory frame. Setting RBP protects the received data, and also tells the main program that there is data to transfer to one of the RSD buffers.

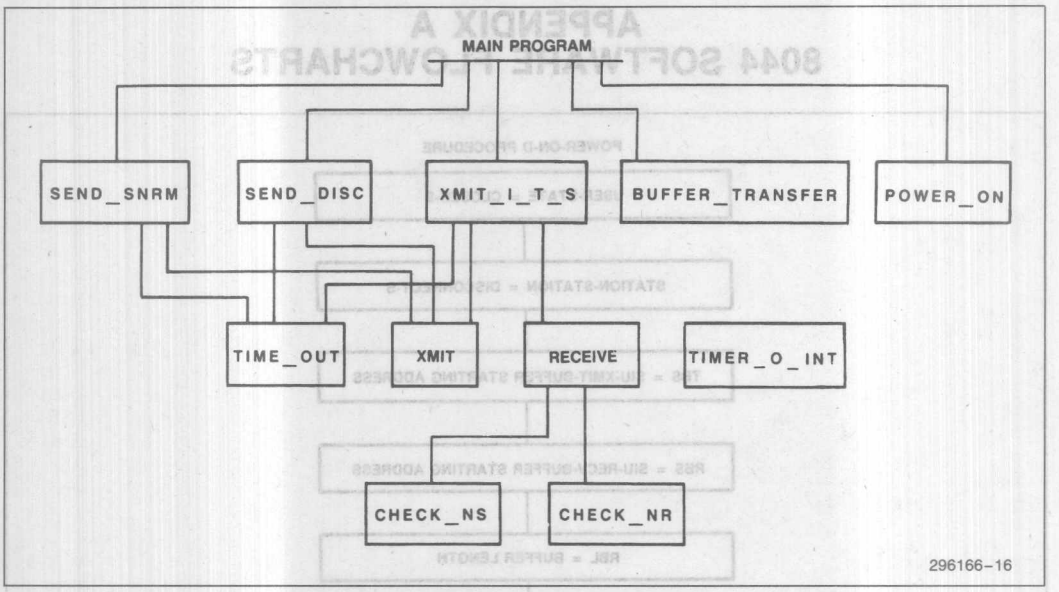


Figure 17. Block Diagram of Primary Station Software Structure

If a supervisory frame is received, the Nr field is checked. If a FALSE is returned, then the station state is set to GO__TO__DISC. If the supervisory frame received was an RNR, buffer status is set to not ready. If the response is not an I frame, nor a supervisory frame, then it must be an Unnumbered frame.

The only Unnumbered frames the primary recognizes are UA, DM, and FRMR. In any event, the station state is set to GO__TO__DISC. However, if the frame received is a FRMR, Nr in the second data byte of the I field is checked to see if the secondary acknowledged an I frame received before it went into the FRMR state. If this is not done and the secondary acknowledged an I frame which the primary did not recognize, the primary transmits the I frame when the secondary returns to the I__T__S. In this case, the secondary would receive duplicate I frames.

APPENDIX A
8044 SOFTWARE FLOWCHARTS

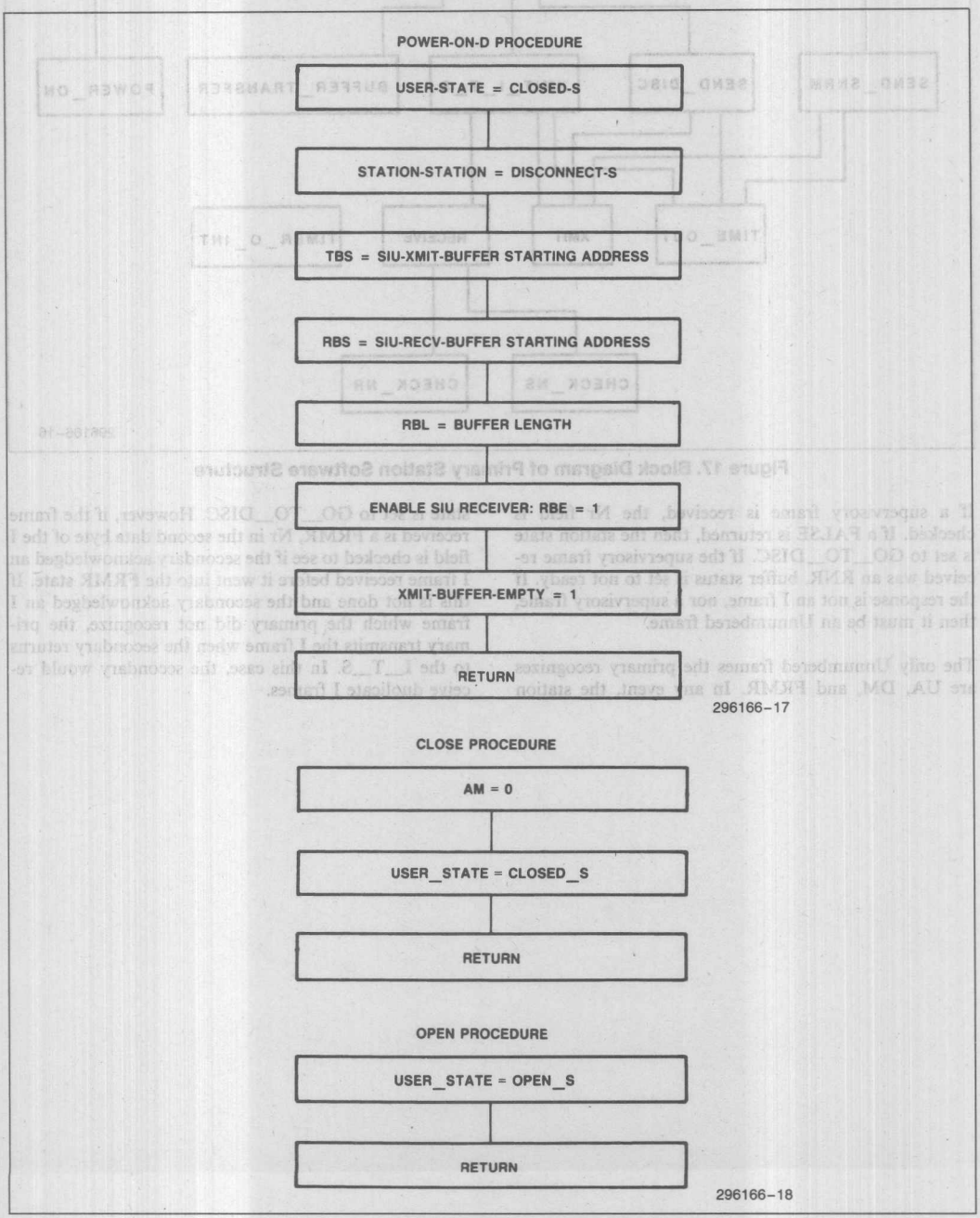


Figure 18. Secondary Station Driver Flow Chart

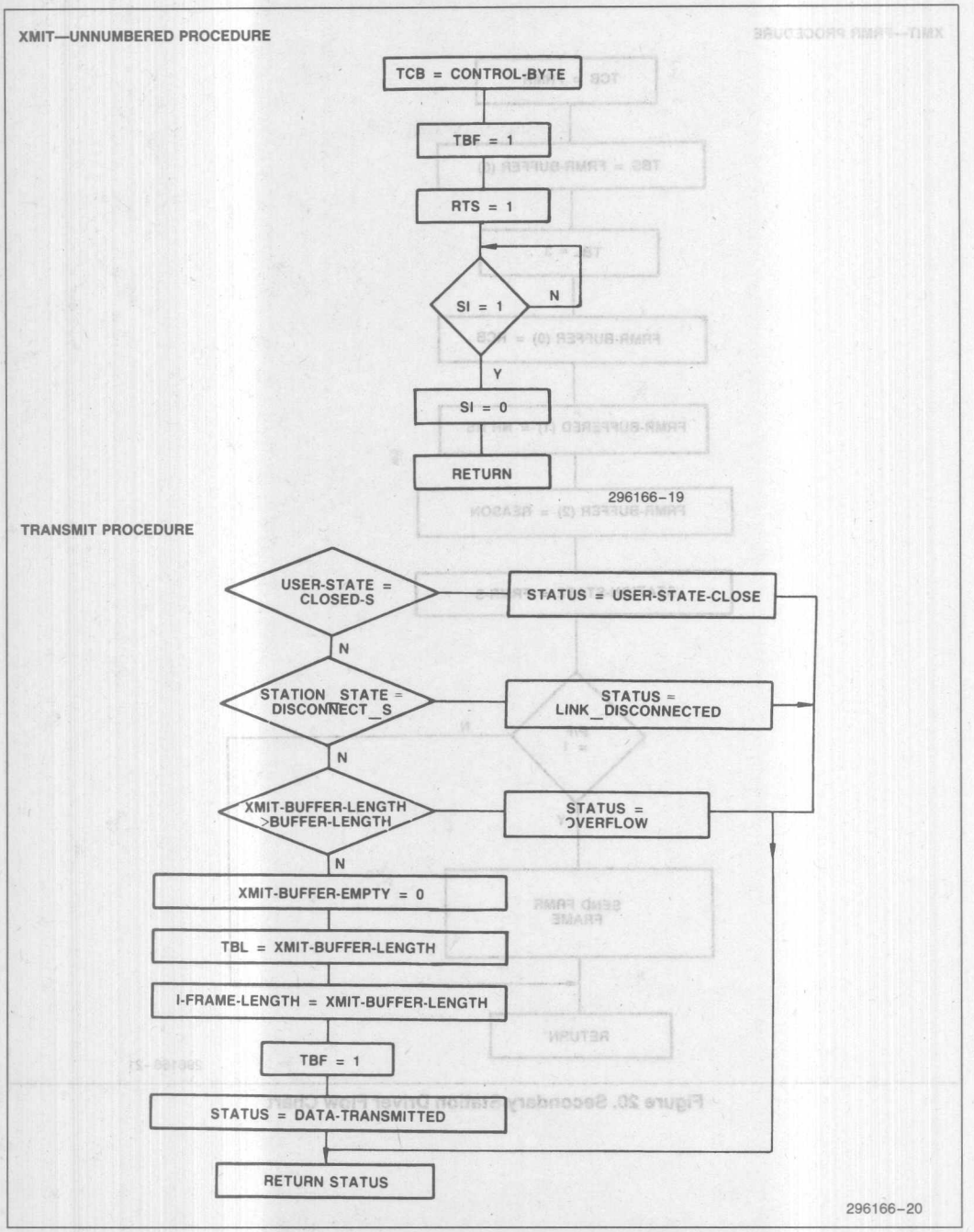


Figure 19. Secondary Station Driver Flow Chart

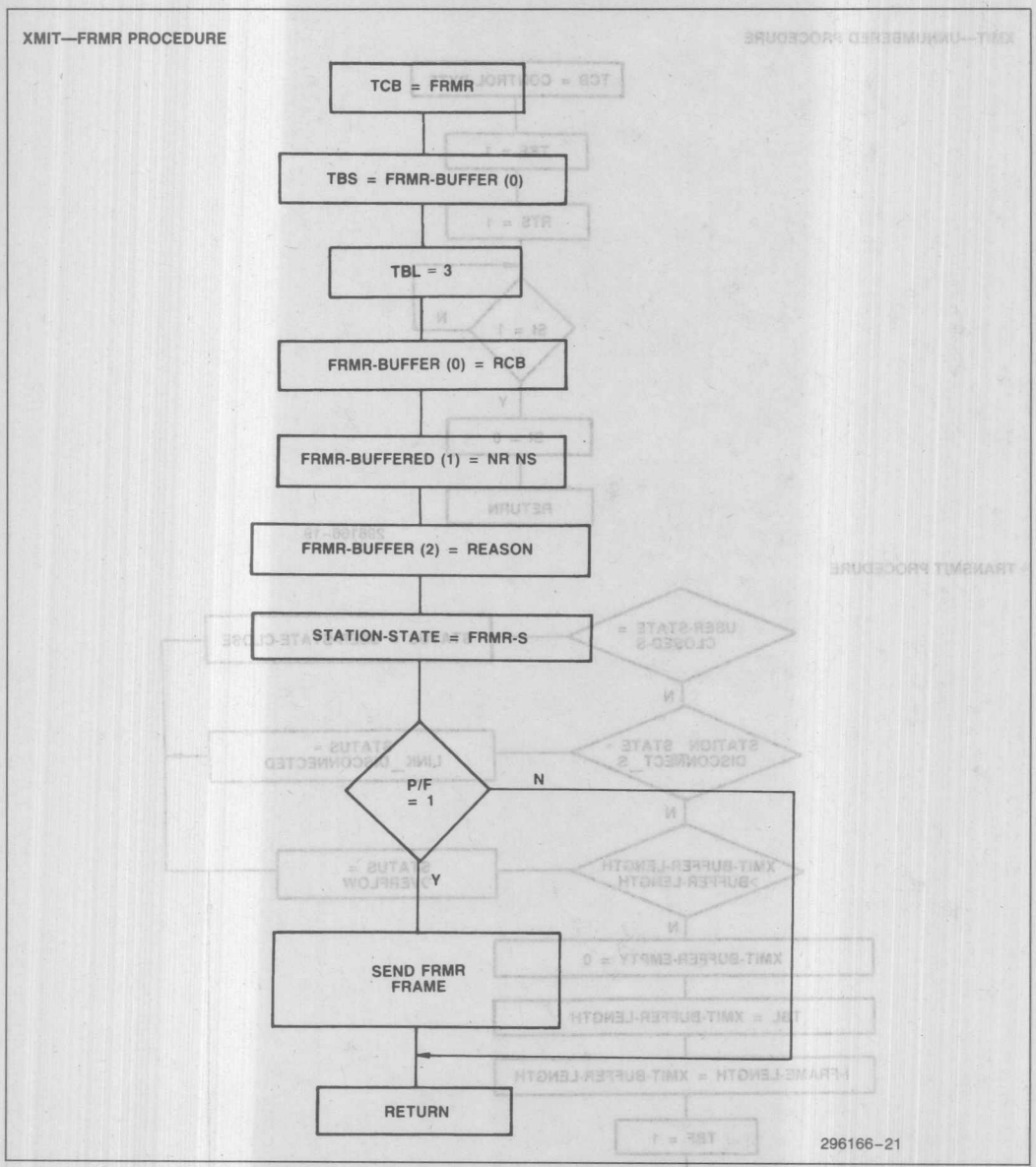


Figure 20. Secondary Station Driver Flow Chart

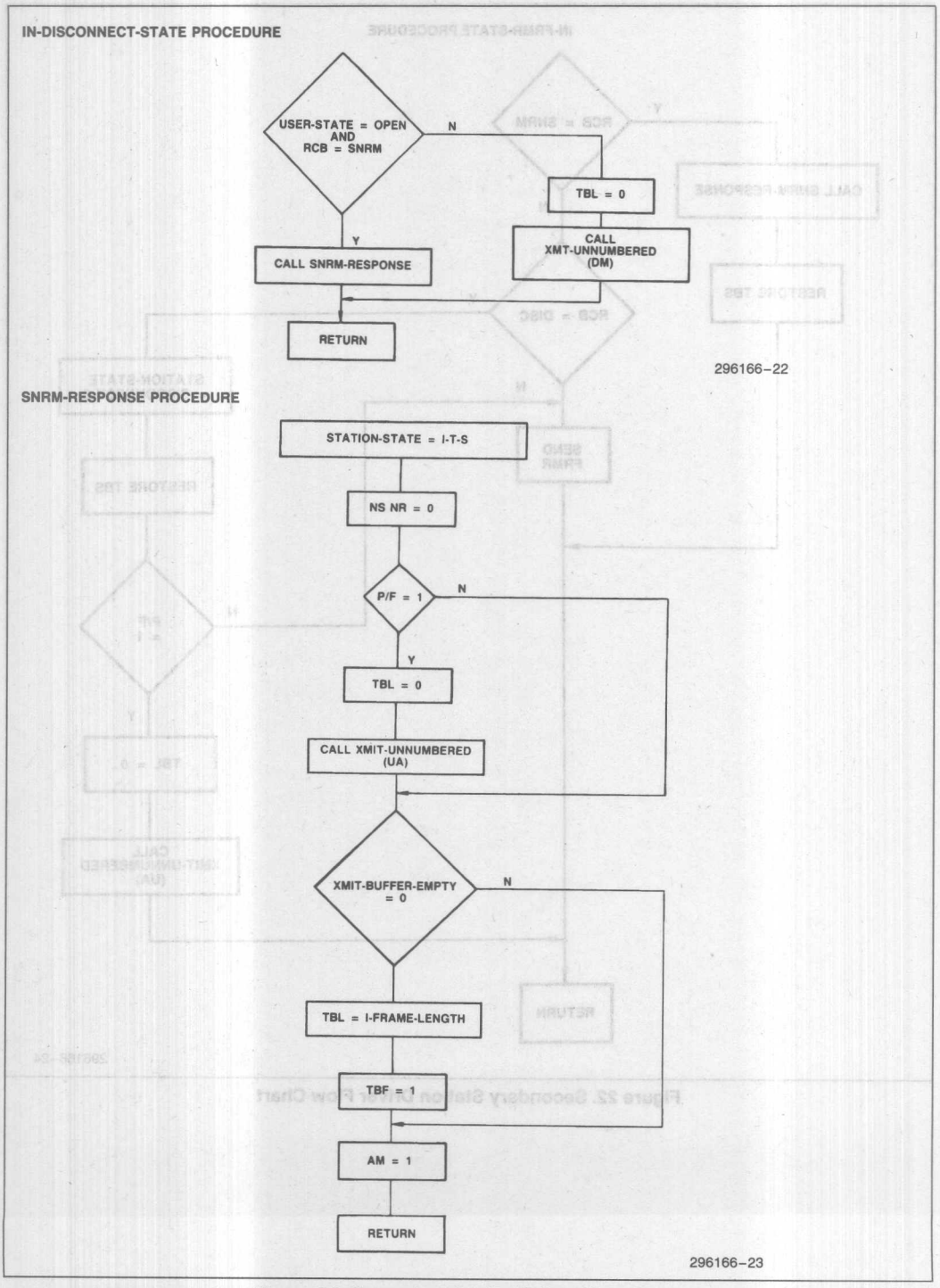


Figure 21. Secondary Station Driver Flow Chart

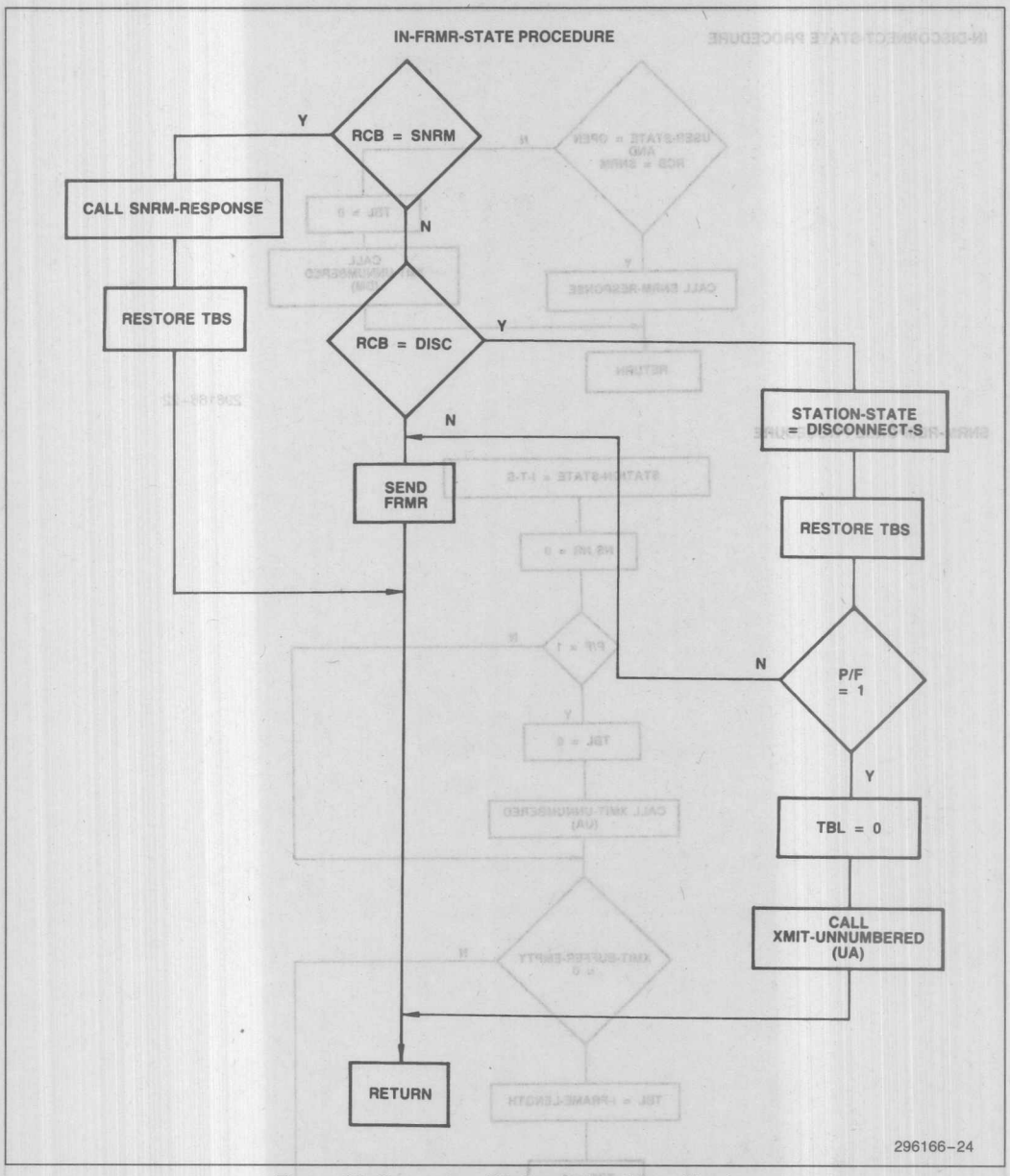


Figure 22. Secondary Station Driver Flow Chart

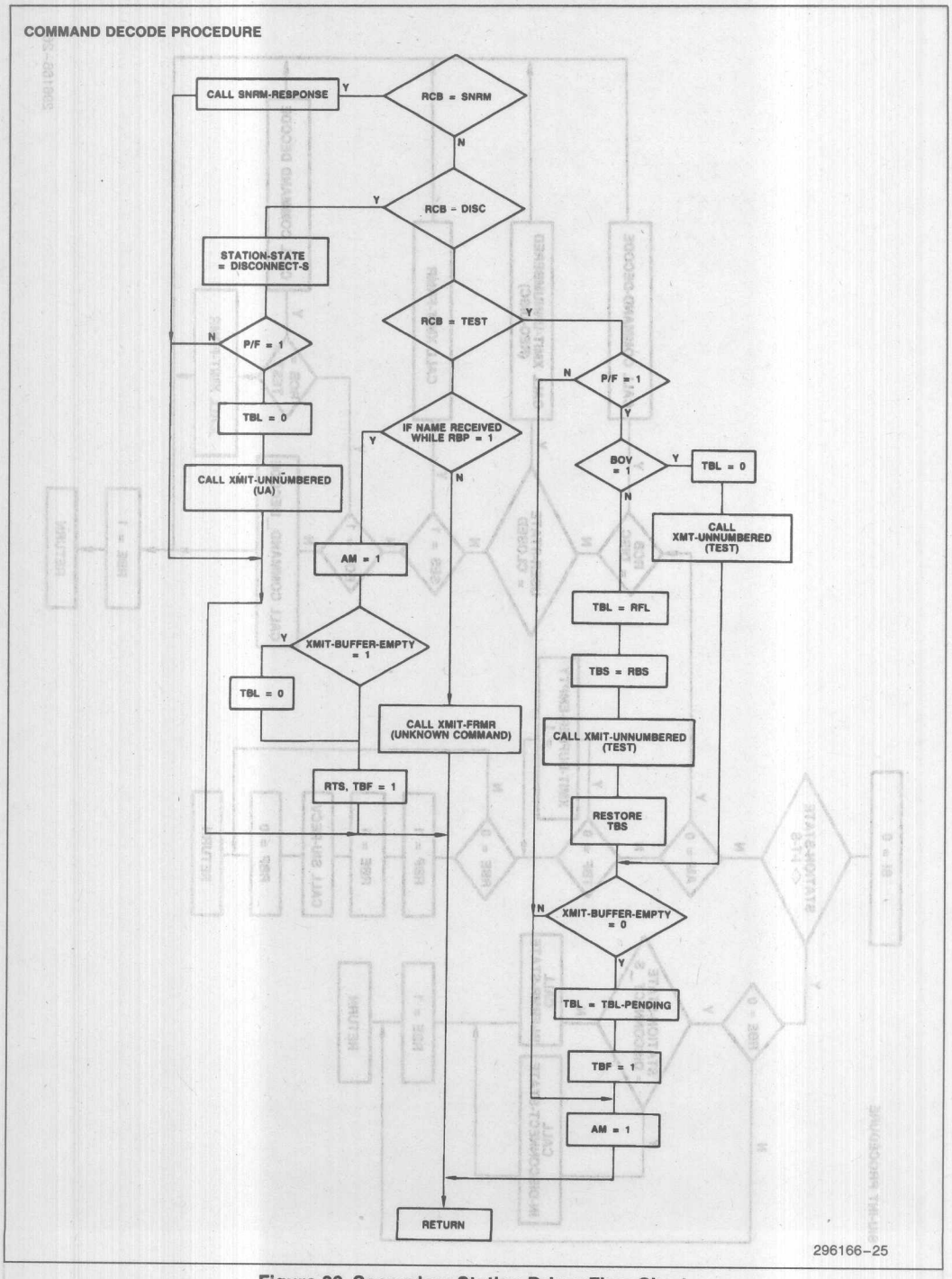


Figure 23. Secondary Station Driver Flow Chart

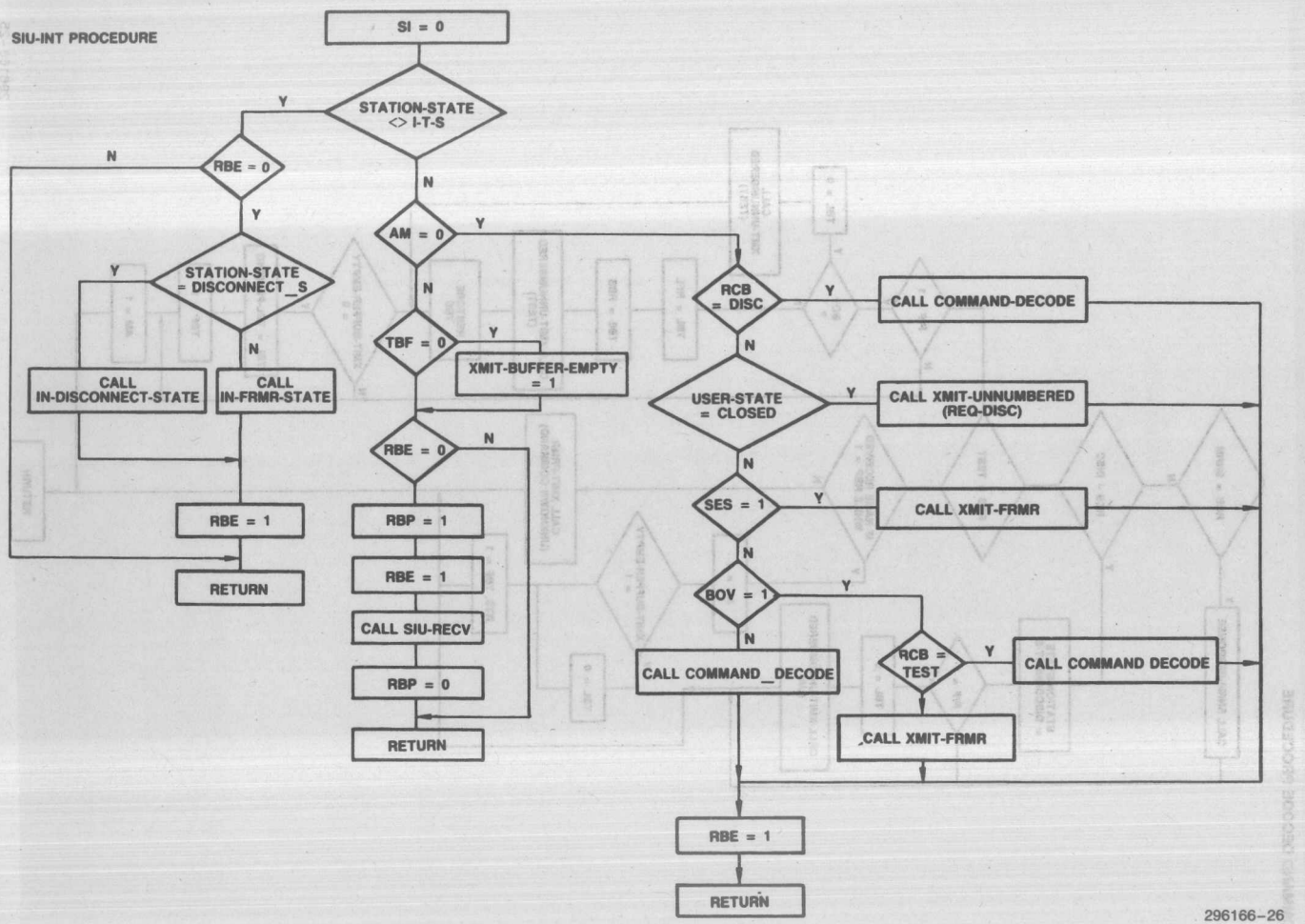


Figure 24. Secondary Station Driver Flow Chart

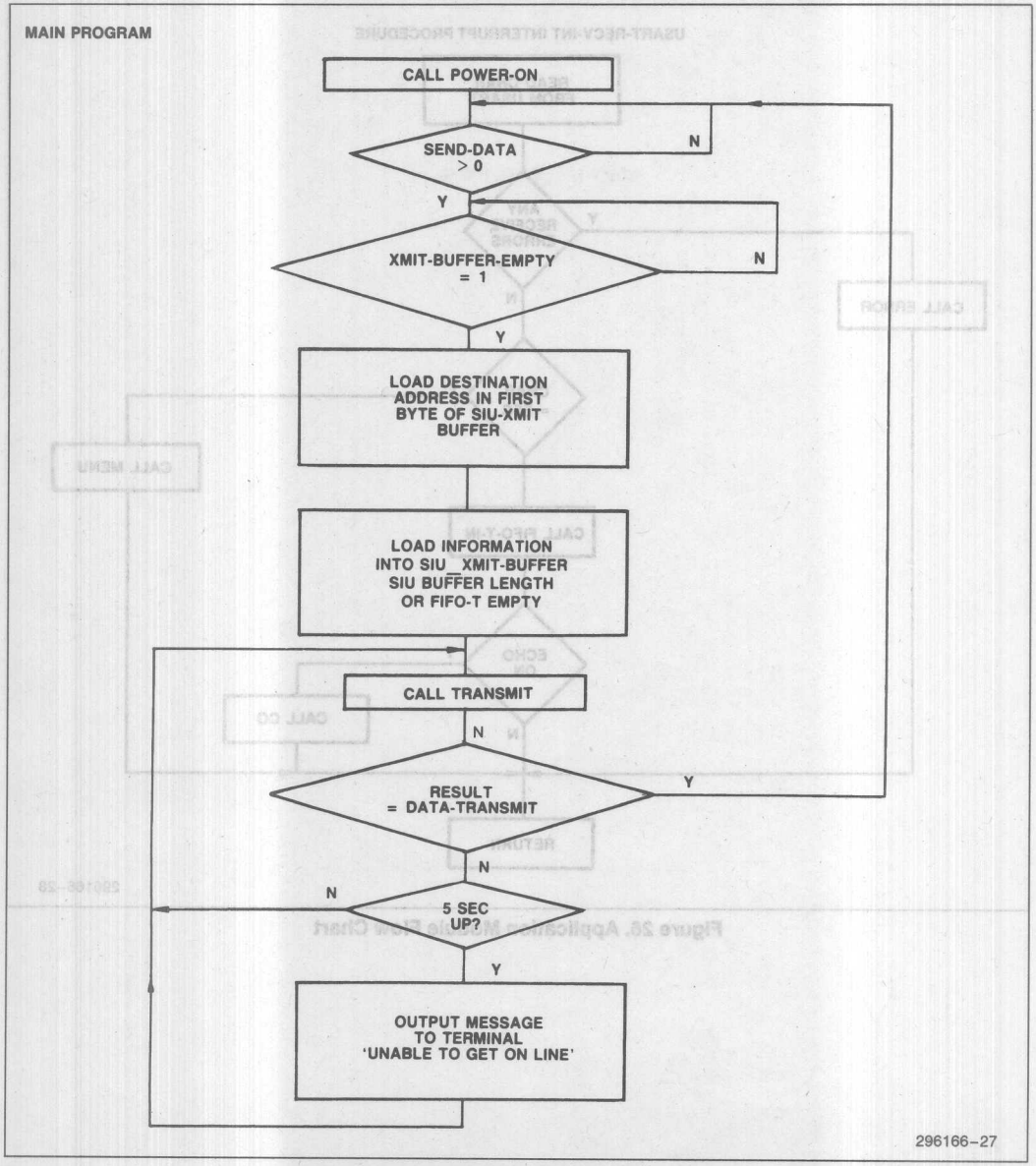


Figure 25. Application Module Flow Chart

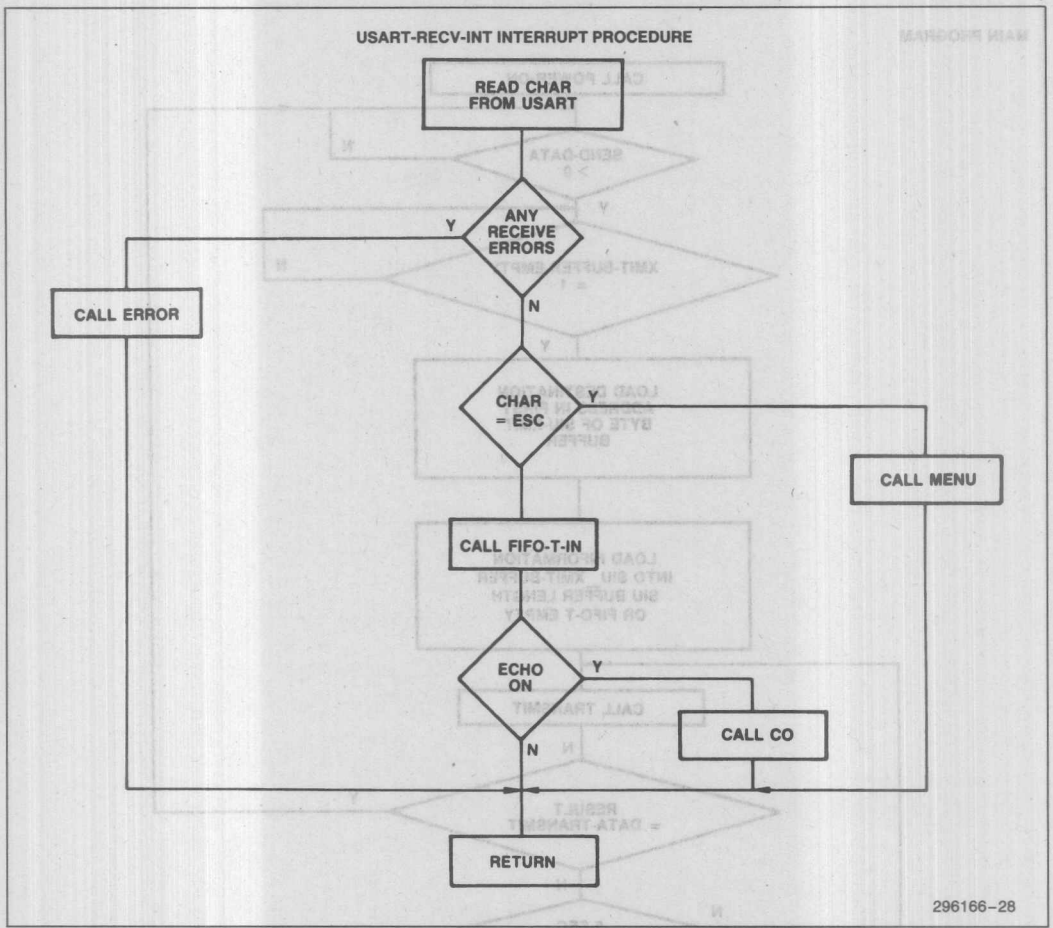


Figure 26. Application Module Flow Chart

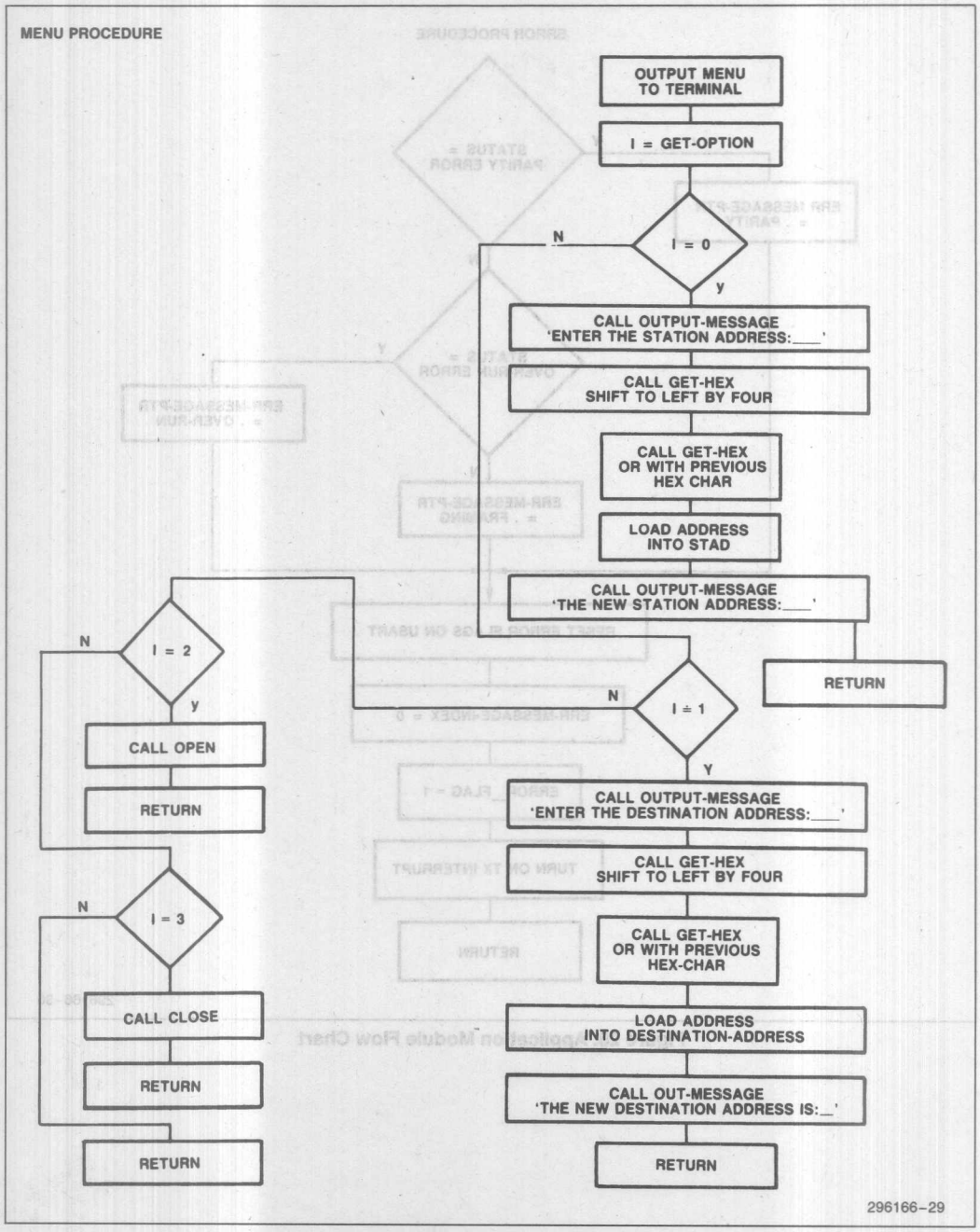


Figure 27. Application Module Flow Chart

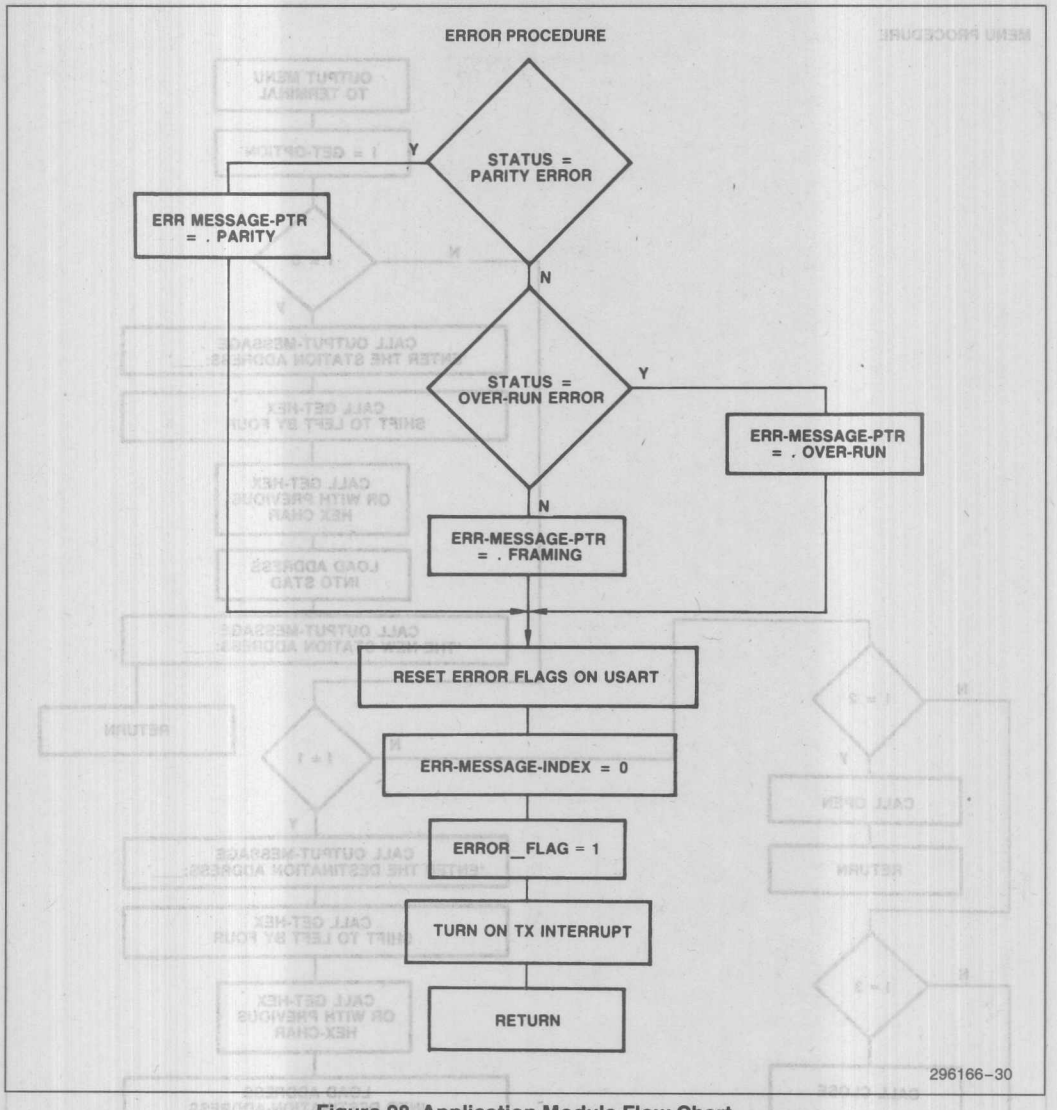


Figure 28. Application Module Flow Chart

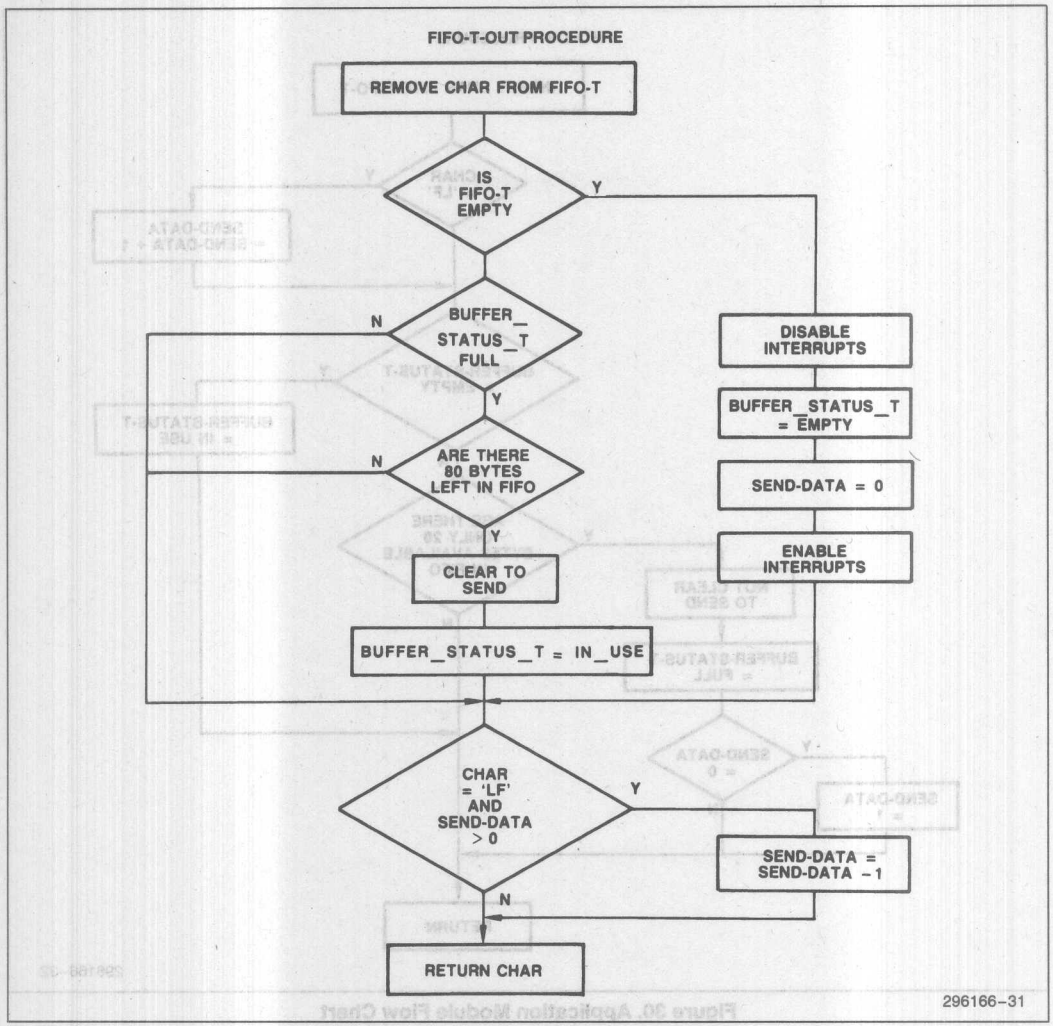


Figure 29. Application Module Flow Chart

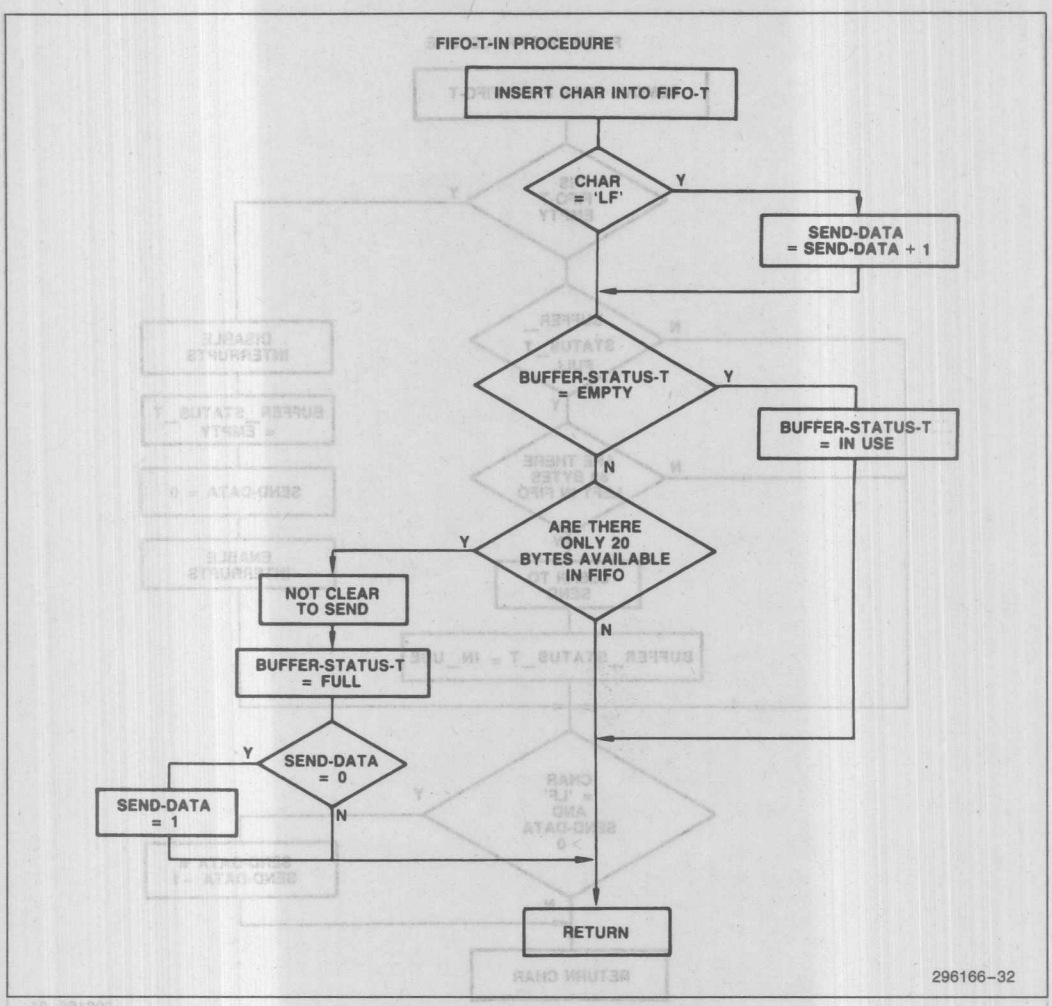


Figure 30. Application Module Flow Chart

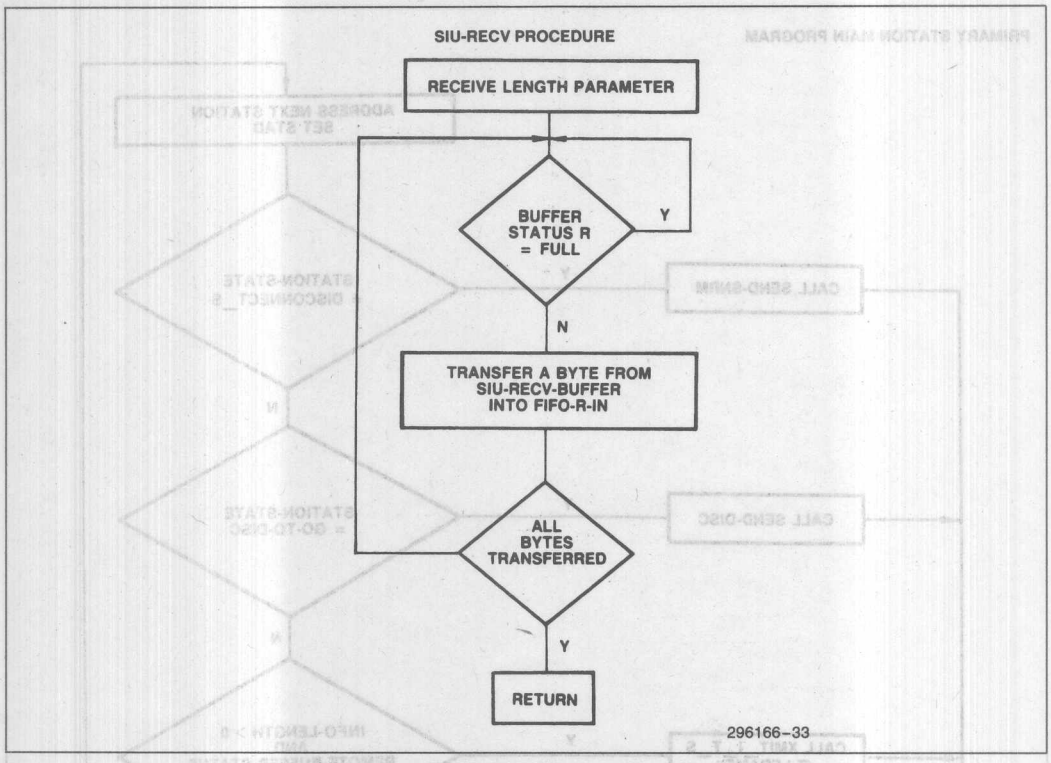


Figure 31. Application Module Flow Chart

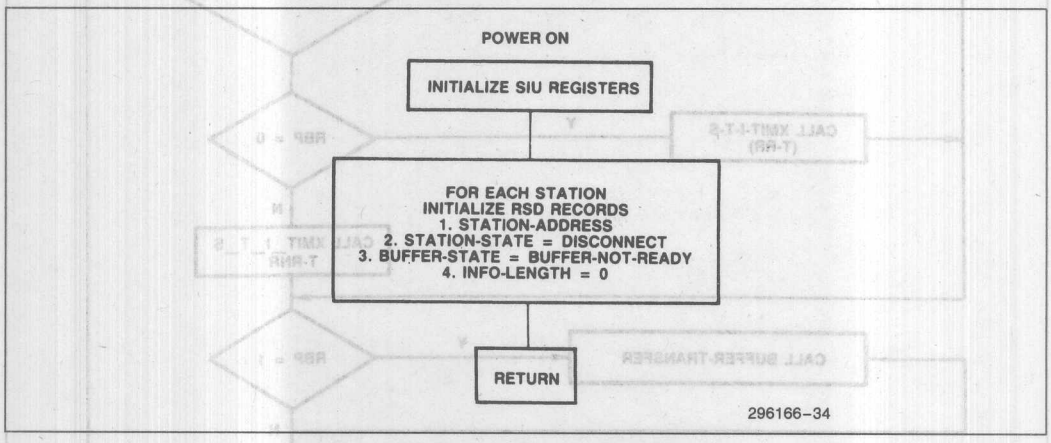


Figure 32. Primary Station Flow Charts

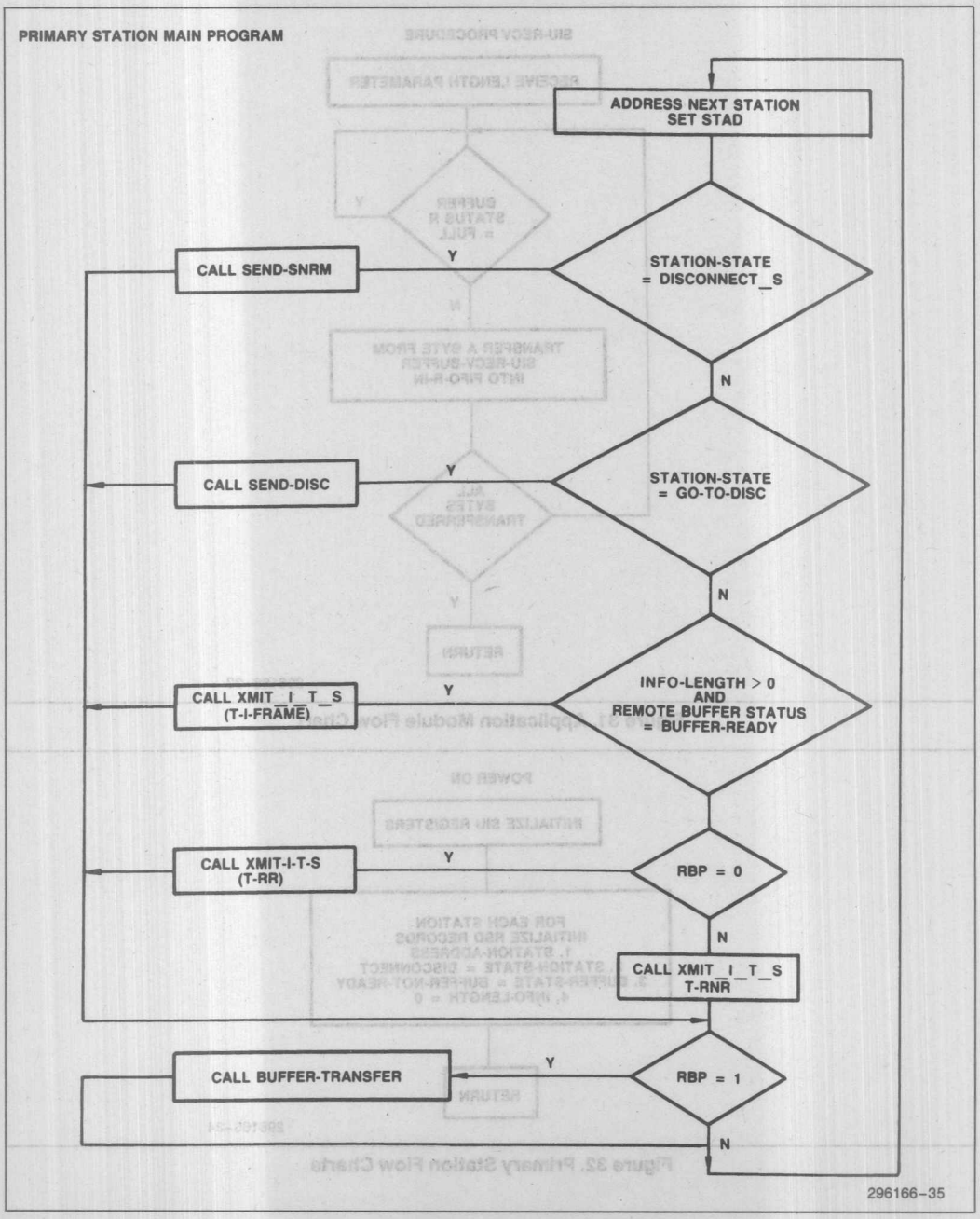


Figure 33. Primary Station Flow Charts

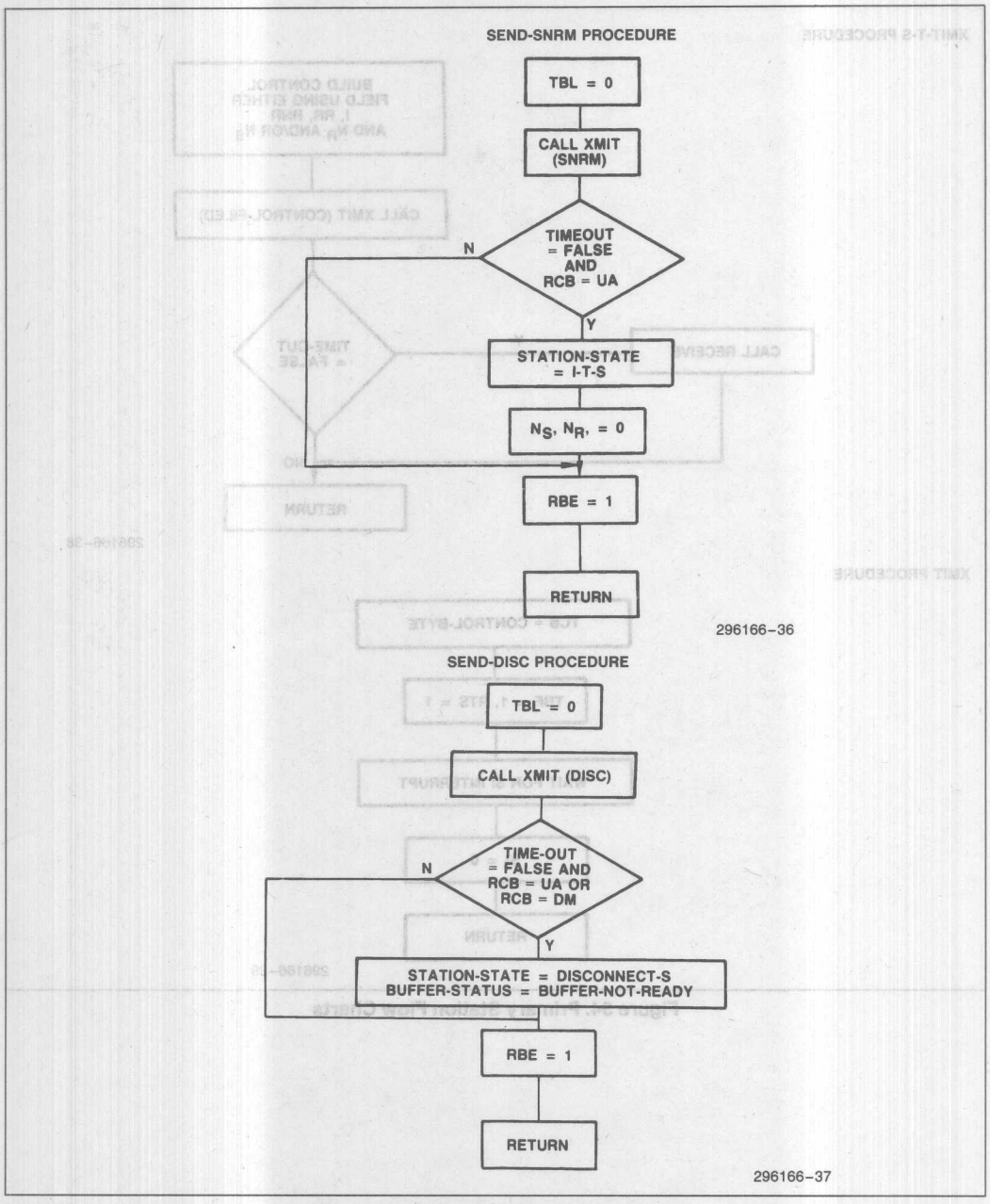


Figure 34. Primary Station Flow Charts

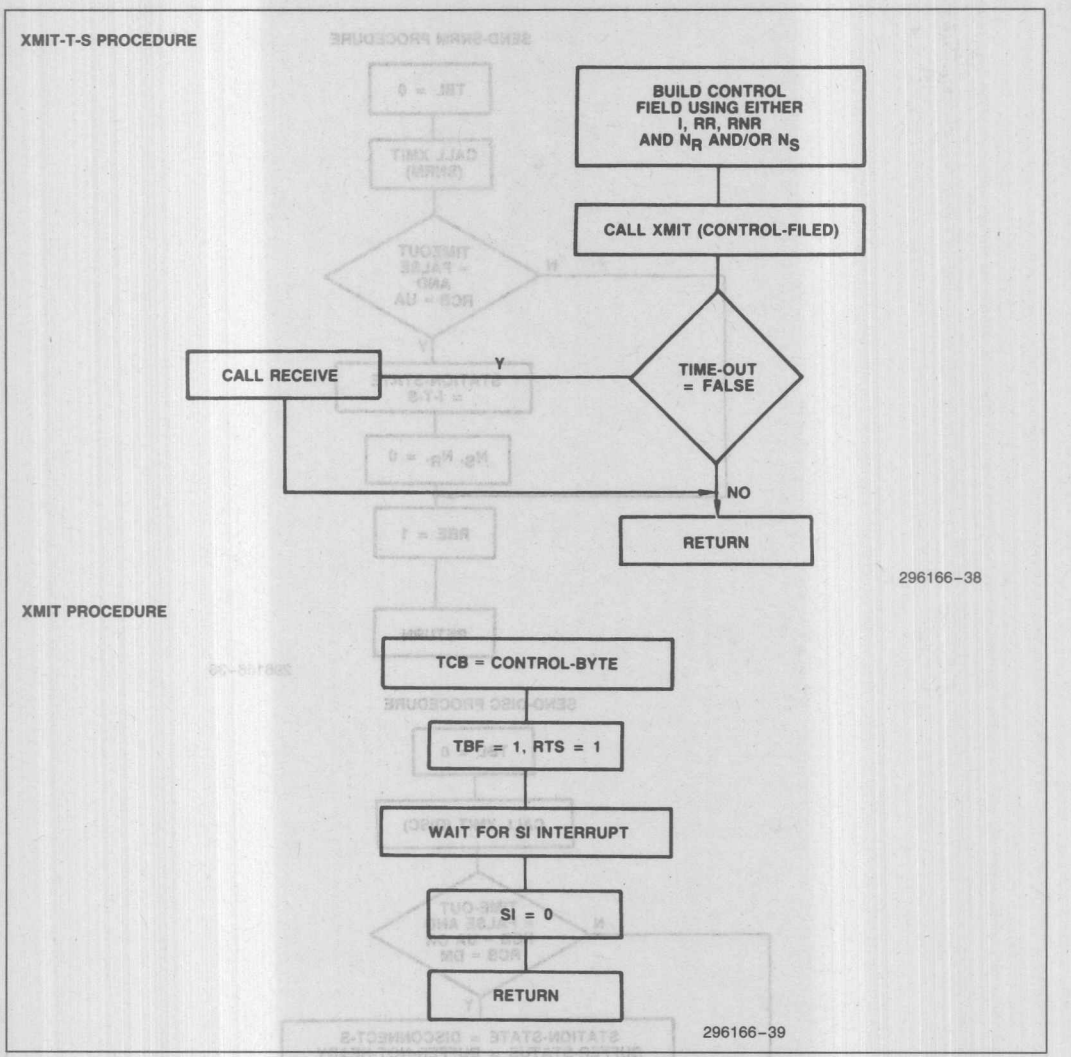


Figure 34. Primary Station Flow Charts

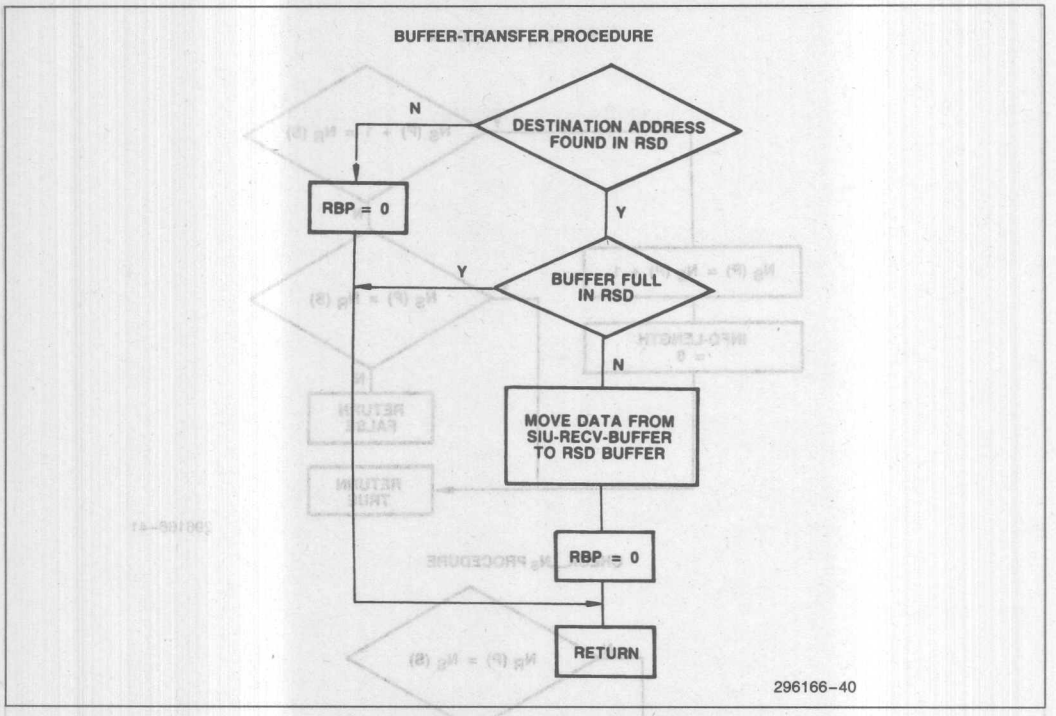


Figure 36. Primary Station Flow Charts

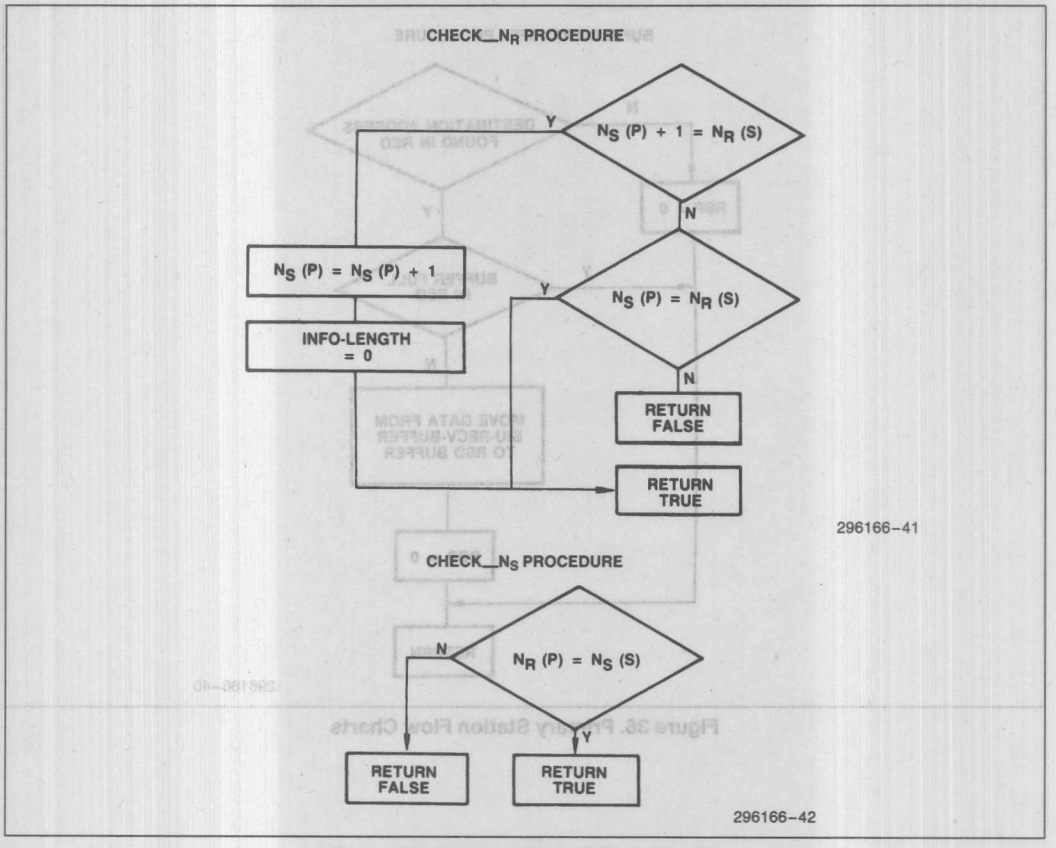


Figure 37. Primary Station Flow Charts

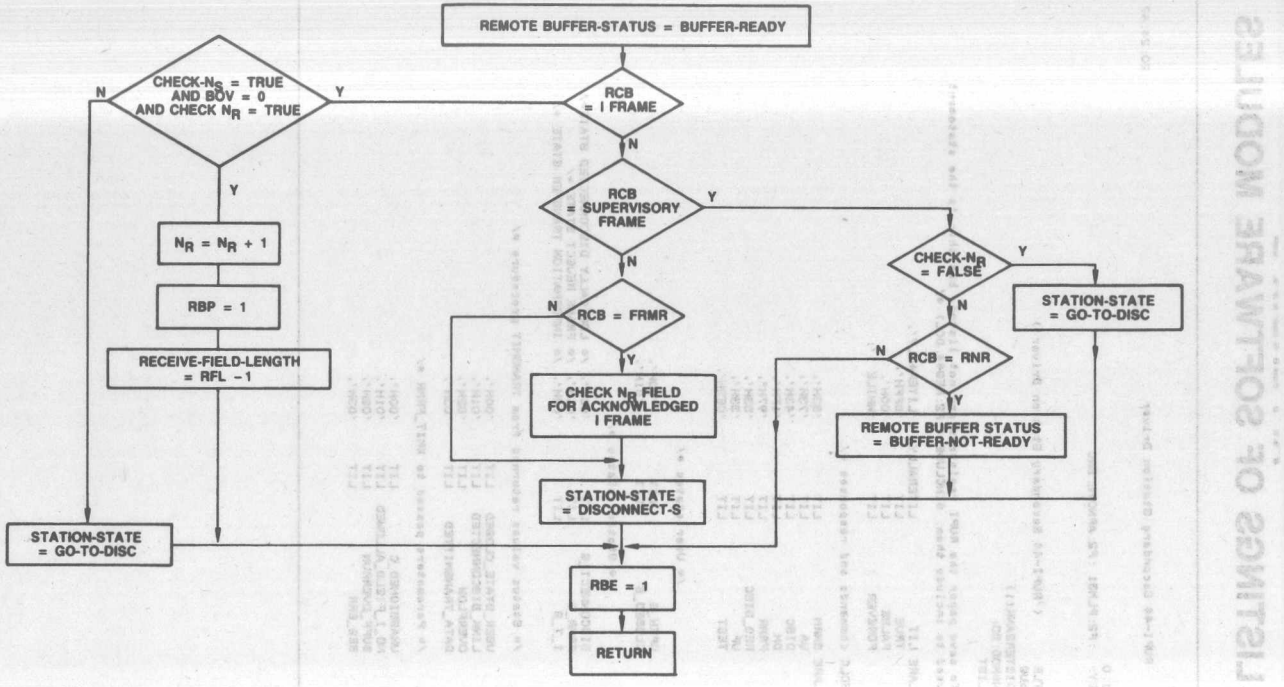


Figure 38. Primary Station Flow Charts

APPENDIX B
LISTINGS OF SOFTWARE MODULES

```
PL/M-51 COMPILER    RUPI-44 Secondary Station Driver                    20:24:47  09/20/83  PAGE   1


ISIS-II PL/M-51 V1.0
COMPILER INVOKED BY:   :F2:PLM51 :F2:APNOTE.SRC


                $TITLE      ('RUPI-44 Secondary Station Driver')
                $DEBUG
                $REGISTERBANK(1)
  1   1         MAIN$MOD:DO;
                $NOLIST

                /* To save paper the RUPI registers are not listed, but this is the statement
                   used to include them: $INCLUDE (:F2:REG44.DCL) */

  5   1         DECLARE LIT             LITERALLY   'LITERALLY',
                        TRUE            LIT         '0FFH',
                        FALSE           LIT         '00H',
                        FOREVER         LIT         'WHILE 1';

                /* SDLC commands and responses */

  6   1         DECLARE SNRM            LIT         '83H',
                        UA              LIT         '73H',
                        DISC            LIT         '43H',
                        DM              LIT         '1FH',
                        FRMR            LIT         '97H',
                        REQ_DISC        LIT         '53H',
                        UP              LIT         '33H',
                        TEST            LIT         '0E3H',

                              /* User states */

                        OPEN_S          LIT         '00H',
                        CLOSED_S        LIT         '01H',

                              /* Station states */

                        DISCONNECT_S    LIT         '00H', /* LOGICALLY DISCONNECTED STATE*/
                        FRMR_S          LIT         '01H', /* FRAME REJECT STATE */
                        I_T_S           LIT         '02H', /* INFORMATION TRANSFER STATE */

                        /* Status values returned from TRANSMIT procedure */

                        USER_STATE_CLOSED   LIT     '00H',
                        LINK_DISCONNECTED   LIT     '01H',
                        OVERFLOW            LIT     '02H',
                        DATA_TRANSMITTED    LIT     '03H',

                        /* Parameters passed to XMIT_FRMR */

                        UNASSIGNED_C        LIT     '00H',
                        NO_I_FIELD_ALLOWED  LIT     '01H',
                        BUFF_OVERRUN        LIT     '02H',
                        SES_ERR             LIT     '03H',
```

296166-44

```
PL/M-51 COMPILER    RUPI-44 Secondary Station Driver                    20:24:47  09/20/83  PAGE   2

                            /* Variables */

                        USER_STATE          BYTE    AUXILIARY,
                        STATION_STATE       BYTE    AUXILIARY,
                        I_FRAME_LENGTH      BYTE    AUXILIARY,

                            /* Buffers */

                        BUFFER_LENGTH                       LIT     '60',
                        SIU_XMIT_BUFFER(BUFFER_LENGTH)      BYTE    PUBLIC    IDATA,
                        SIU_RECV_BUFFER(BUFFER_LENGTH)      BYTE    PUBLIC,
                        FRMR_BUFFER(3)              BYTE,

                            /* Flags */

                        XMIT_BUFFER_EMPTY       BIT PUBLIC;

  7   2     SIU_RECV: PROCEDURE (LENGTH) EXTERNAL;
  8   2         DECLARE LENGTH  BYTE;
  9   1     END SIU_RECV;

 10   2     OPEN: PROCEDURE PUBLIC USING 2;
 11   2             USER_STATE=OPEN_S;
 12   1     END OPEN;

 13   2     CLOSE: PROCEDURE PUBLIC USING 2;
 14   2             AM=0;
 15   2             USER_STATE=CLOSED_S;
 16   1     END CLOSE;

 17   2     POWER_ON_D: PROCEDURE   PUBLIC USING 0;

 18   2             USER_STATE=CLOSED_S;
 19   2             STATION_STATE=DISCONNECT_S;
 20   2             TBS=.SIU_XMIT_BUFFER(0);
 21   2             RBS=.SIU_RECV_BUFFER(0);
 22   2             RBL=BUFFER_LENGTH;
 23   2             RBE=1;          /* Enable the SIU's receiver */
 24   2             XMIT_BUFFER_EMPTY=1;

 25   1     END POWER_ON_D;

 26   2     TRANSMIT: PROCEDURE (XMIT_BUFFER_LENGTH) BYTE   PUBLIC USING 0;

           /* User must check XMIT_BUFFER_EMPTY flag before calling this procedure */

 27   2             DECLARE XMIT_BUFFER_LENGTH  BYTE,
                             I                   BYTE    AUXILIARY,
                             STATUS              BYTE    AUXILIARY;

 28   2             IF USER_STATE=CLOSED_S
                        THEN STATUS=USER_STATE_CLOSED;
 30   2             ELSE IF STATION_STATE=DISCONNECT_S
                        THEN STATUS=LINK_DISCONNECTED;
 32   2             ELSE IF XMIT_BUFFER_LENGTH>BUFFER_LENGTH
                        THEN STATUS=OVERFLOW;
 34   3             ELSE DO;
```

296166-45

```
PL/M-51 COMPILER    RUPI-44 Secondary Station Driver                20:24:47  09/20/83  PAGE   3

  35   3                    XMIT_BUFFER_EMPTY=0;
  36   3                    TBL=XMIT_BUFFER_LENGTH;
  37   3                    I_FRAME_LENGTH=XMIT_BUFFER_LENGTH; /* Store length in case station
                                                                 is reset by FRMR, SNRM etc. */
  38   3                    TBF=1;
  39   3                    STATUS=DATA_TRANSMITTED;
  40   3                  END;
  41   2              RETURN STATUS;
  42   1      END TRANSMIT;

  43   2      XMIT_UNNUMBERED: PROCEDURE (CONTROL_BYTE) ;

  44   2          DECLARE CONTROL_BYTE    BYTE;

  45   2              TCB=CONTROL_BYTE;
  46   2              TBF=1;
  47   2              RTS=1;
  48   3              DO WHILE NOT SI;
  49   3              END;
  50   2              SI=0;

  51   1      END XMIT_UNNUMBERED;

  52   2      SNRM_RESPONSE: PROCEDURE ;

  53   2          STATION_STATE=I_T_S;
  54   2          NSNR=0;
  55   2          IF (RCB AND 10H) <> 0 /* Respond if polled */
                     THEN DO;
  57   3                    TBL=0;
  58   3                    CALL XMIT_UNNUMBERED(UA);
  59   3                END;
  60   2          IF XMIT_BUFFER_EMPTY=0  /* If an I frame was left pending transmission
                                             then restore it */
              THEN DO;
  62   3                    TBL=I_FRAME_LENGTH;
  63   3                    TBF=1;
  64   3                END;
  65   2          AM=1;

  66   1      END SNRM_RESPONSE;

  67   2      XMIT_FRMR: PROCEDURE (REASON)  ;

  68   2              DECLARE REASON  BYTE;

  69   2              TCB=FRMR;

  70   2              TBS=.FRMR_BUFFER(0);
  71   2              TBL=3;
  72   2              FRMR_BUFFER(0)=RCB;
                      /* Swap nibbles in NSNR */
  73   2              FRMR_BUFFER(1)=(SHL((NSNR AND 0EH),4) OR  SHR((NSNR AND 0E0H),4));
  74   3              DO CASE REASON;
  75   3                  FRMR_BUFFER(2)=01H;   /* UNASSIGNED_C */
```

296166-46

```
PL/M-51 COMPILER    RUPI-44 Secondary Station Driver                    20:24:47  09/20/83  PAGE   4

   76   3                FRMR_BUFFER(2)=02H;   /* NO_I_FIELD_ALLOWED */
   77   3                FRMR_BUFFER(2)=04H;   /* BUFF_OVERRUN */
   78   3                FRMR_BUFFER(2)=08H;   /* SES_ERR */
   79   3             END;

   80   2             STATION_STATE=FRMR_S;

   81   2             IF (RCB AND 10H) <>0
                        THEN DO;
   83   3                     TBF=1;
   84   3                     RTS=1;
   85   4                     DO WHILE NOT SI;
   86   4                     END;
   87   3                     SI=0;
   88   3                  END;
   89   1      END XMIT_FRMR;


   90   2      IN_DISCONNECT_STATE: PROCEDURE ;/* Called from SIU_INT procedure */

   91   2              IF ((USER_STATE=OPEN_S) AND ((RCB AND OEFH)=SNRM))
                          THEN CALL SNRM_RESPONSE;

   93   2              ELSE IF (RCB AND 10H) <> 0
                           THEN DO;
   95   3                        TBL=0;
   96   3                        CALL XMIT_UNNUMBERED(DM);
   97   3                     END;
   98   1      END IN_DISCONNECT_STATE;


   99   2      IN_FRMR_STATE: PROCEDURE ; /* Called by SIU_INT when a frame has been received
                                            when in the FRMR state */

  100   2          IF (RCB AND OEFH)=SNRM
                    THEN DO;
  102   3                 CALL SNRM_RESPONSE;
  103   3                 TBS=.SIU_XMIT_BUFFER(0);  /* Restore transmit buffer start address */
  104   3              END;


  105   2          ELSE IF (RCB AND OEFH)=DISC
                        THEN DO;
  107   3                 STATION_STATE=DISCONNECT_S;
  108   3                 TBS=.SIU_XMIT_BUFFER(0);   /* Restore transmit buffer start address */
  109   3                 IF (RCB AND 10H)<> 0
                             THEN DO;
  111   4                        TBL=0;
  112   4                        CALL XMIT_UNNUMBERED(UA);
  113   4                     END;
  114   3              END;

  115   3          ELSE DO;       /* Receive control byte is something other than DISC or SNRM */
  116   3                 IF (RCB AND 10H) <> 0
                             THEN DO;
  118   4                        TBF=1;
  119   4                        RTS=1;
```

296166-47

```
PL/M-51 COMPILER    RUPI-44 Secondary Station Driver                    20:24:47  09/20/83  PAGE   5

 120   5                          DO WHILE NOT SI;
 121   5                          END;
 122   4                        END;
 123   3              END;

 124   1      END IN_FRMR_STATE;

 125   2      COMMAND_DECODE: PROCEDURE ;


 126   2          IF (RCB AND OEFH)=SNRM
                     THEN  CALL SNRM_RESPONSE;

 128   2          ELSE IF (RCB AND OEFH)=DISC
                          THEN  DO;
 130   3                              STATION_STATE=DISCONNECT_S;
 131   3                              IF (RCB AND 10H)<>0
                                         THEN DO;
 133   4                                       TBL=0;
 134   4                                       CALL XMIT_UNNUMBERED(UA);
 135   4                                     END;
 136   3                        END;

 137   2          ELSE IF (RCB AND OEFH)=TEST
                          THEN DO;
 139   3                              IF (RCB AND 10H)>0  /* Respond if polled */
                                         THEN DO;     /* FOR BOV=1, SEND THE TEST RESPONSE WITHOUT AN I FIELD */
 141   4                                       IF (BOV=1)
                                                 THEN DO;
 143   5                                               TBL=0;
 144   5                                               CALL XMIT_UNNUMBERED(TEST OR 10H);
 145   5                                             END;
 146   5                                       ELSE DO; /* If no BOV, send received I field back to primary */
 147   5                                               TBL=RFL;
 148   5                                               TBS=RBS;
 149   5                                               CALL XMIT_UNNUMBERED(TEST OR 10H);
 150   5                                               TBS=.SIU_XMIT_BUFFER(0);  /* Restore TBS */
 151   5                                             END;

                                                 /* If an I frame was pending, set it up again */

 152   4                                       IF XMIT_BUFFER_EMPTY=0
                                                     THEN DO;
 154   5                                                   TBL=I_FRAME_LENGTH;
 155   5                                                   TBF=1;
 156   5                                                 END;
 157   4                                     END;
 158   3                              AM=1;
 159   3                            END;

 160   2          ELSE IF (RCB AND 01H) = 0 /* Kicked out of the AUTO mode because
                                               an I frame was received while RPB = 1 */
                          THEN DO;
 162   3                              AM = 1;
 163   3                              IF XMIT_BUFFER_EMPTY = 1
                                         THEN TBL = 0;
 165   3                              TBF = 1;    /* Send an AUTO mode response */
```

296166-48

```
PL/M-51 COMPILER    RUPI-44 Secondary Station Driver                    20:24:47  09/20/83  PAGE   6


 166   3                      RTS = 1;
 167   3                    END;

 168   2          ELSE CALL XMIT_FRMR(UNASSIGNED_C);  /* Received an undefined or not implemented command */

 169   1      END COMMAND_DECODE;


 170   2      SIU_INT: PROCEDURE INTERRUPT 4;

 171   2              DECLARE     I   BYTE    AUXILIARY;

 172   2              SI=0;
 173   2              IF STATION_STATE<> I_T_S /* Must be in NON-AUTO mode */
                         THEN DO;
 175   3                      IF RBE=0 /* Received a frame?  Give response */
                                 THEN DO;
 177   3                                  DO CASE STATION_STATE;
 178   5                                      CALL IN_DISCONNECT_STATE;
 179   5                                      CALL IN_FRMR_STATE;
 180   3                                  END;
 181   4                                  RBE=1;
 182   4                              END;
 183   3                      RETURN;
 184   3                      END;

                         /* If the program reaches this point, STATION_STATE=I_T_S
                             which means the SIU either was, or still is in the AUTO MODE */


 185   2              IF AM=0
                         THEN DO;
 187   3                      IF (RCB AND 0EFH)=DISC
                                 THEN CALL COMMAND_DECODE;
 189   3                      ELSE IF USER_STATE=CLOSED_S
                                 THEN DO;
 191   4                                  TBL=0;
 192   4                                  CALL XMIT_UNNUMBERED(REQ_DISC);
 193   4                              END;
 194   3                      ELSE IF SES=1
                                 THEN CALL XMIT_FRMR(SES_ERR);
 196   3                      ELSE IF BOV=1
                                 THEN DO; /* DON'T SEND FRMR IF A TEST WAS RECEIVED*/
 198   4                                  IF (RCB AND 0EFH)=TEST
                                             THEN CALL COMMAND_DECODE;
 200   4                                  ELSE     CALL XMIT_FRMR(BUFF_OVERRUN);
 201   4                              END;
 202   3                      ELSE CALL COMMAND_DECODE;
 203   3                      RBE=1;
 204   3                  END;

 205   3              ELSE DO; /* MUST STILL BE IN AUTO MODE */
 206   3                      IF TBF=0
                                 THEN XMIT_BUFFER_EMPTY=1;  /* TRANSMITTED A FRAME */
 208   3                      IF RBE=0
                                 THEN DO;
```

296166-49

```
PL/M-51 COMPILER    RUPI-44 Secondary Station Driver                    20:24:47  09/20/83  PAGE   7


 210   4                                  RBP=1; /* RNR STATE */
 211   4                                  RBE=1; /* RE-ENABLE RECEIVER */
 212   4                                  CALL SIU_RECV(RFL);
 213   4                                  RBP=0; /* RR STATE */
 214   4                              END;
 215   3                  END;
 216   1      END SIU_INT;

 217   1      END MAIN0MOD;

Software and application note written by Charles Yager


WARNINGS:
    4 IS THE HIGHEST USED INTERRUPT


MODULE INFORMATION:                 (STATIC+OVERLAYABLE)
    CODE SIZE                       = 028FH        655D
    CONSTANT SIZE                   = 0000H          0D
    DIRECT VARIABLE SIZE            =   3FH+02H     63D+  2D
    INDIRECT VARIABLE SIZE          =   3CH+00H     60D+  0D
    BIT SIZE                        =   01H+00H      1D+  0D
    BIT-ADDRESSABLE SIZE            =   00H+00H      0D+  0D
    AUXILIARY VARIABLE SIZE         = 0006H          6D
    MAXIMUM STACK SIZE              = 0017H         23D
    REGISTER-BANK(S) USED:            0 1 2
    460 LINES READ
    0 PROGRAM ERROR(S)
END OF PL/M-51 COMPILATION
```

296166-50

```
PL/M-51 COMPILER    Application Module: Async/SDLC Protocol converter                18:30:53  09/19/83  PAGE   1


ISIS-II PL/M-51 V1.0
COMPILER INVOKED BY:  :f2:plm51 :f2:unote.src


                $TITLE      ('Application Module: Async/SDLC Protocol converter')
                $debug
                $registerbank(0)
  1   1         user$mod:do;
                $NOLIST
  5   1         DECLARE     LIT                 LITERALLY       'LITERALLY',
                            TRUE                LIT             '0FFH',
                            FALSE               LIT             '00H',
                            FOREVER             LIT             'WHILE 1',
                            ESC                 LIT             '1BH',
                            LF                  LIT             '0AH',
                            CR                  LIT             '0DH',
                            BS                  LIT             '08H',
                            BEL                 LIT             '07H',
                            EMPTY               LIT             '00H',
                            INUSE               LIT             '01H',
                            FULL                LIT             '02H',
                            USER_STATE_CLOSED   LIT             '00H',
                            LINK_DISCONNECTED   LIT             '01H',
                            OVERFLOW            LIT             '02H',
                            DATA_TRANSMITTED    LIT             '03H',

                               /* BUFFERS */

                            BUFFER_LENGTH       LIT             '60',
                            SIU_XMIT_BUFFER(BUFFER_LENGTH)      BYTE    EXTERNAL    IDATA,
                            SIU_RECV_BUFFER(BUFFER_LENGTH)      BYTE    EXTERNAL,
                            FIFO_T(256)             BYTE    AUXILIARY,
                            IN_PTR_T                BYTE    AUXILIARY,
                            OUT_PTR_T               BYTE    AUXILIARY,
                            BUFFER_STATUS_T         BYTE    AUXILIARY,
                            FIFO_R(256)             BYTE    AUXILIARY,
                            IN_PTR_R                BYTE    AUXILIARY,
                            OUT_PTR_R               BYTE    AUXILIARY,
                            BUFFER_STATUS_R         BYTE    AUXILIARY,

                               /* Variables and Parameters */

                            LENGTH                  BYTE    AUXILIARY,
                            CHAR                    BYTE    AUXILIARY,
                            I                       BYTE    AUXILIARY,
                            USART_CMD               BYTE    AUXILIARY,
                            DESTINATION_ADDRESS     BYTE    AUXILIARY,
                            SEND_DATA               BYTE    AUXILIARY,
                            RESULT                  BYTE    AUXILIARY,
                            ERR_MESSAGE_INDEX       BYTE    AUXILIARY,
                            ERR_MESSAGE_PTR         WORD    AUXILIARY,

                              /*   Messages Sent to the Terminal  */

                            PARITY(*) BYTE CONSTANT(LF,CR,'Parity Error Detected',LF,CR,00H),
                            FRAME(*) BYTE CONSTANT(LF,CR,'Framing Error Detected',LF,CR,00H),
```

296166-51

```
PL/M-51 COMPILER    Application Module: Async/SDLC Protocol converter          18:50:53  09/19/83  PAGE   2


                    OVER_RUN(*) BYTE CONSTANT(LF,CR,'Overrun Error Detected',LF,CR,0),
                    LINK(*) BYTE CONSTANT(LF,CR,'Unable to Get Online',LF,CR,00H),
                    DEST_ADDR(*) BYTE CONSTANT(CR,LF,LF,
                              'Enter the destination address: __', BS, BS, 0),

                    D_ADDR_ACK(*) BYTE CONSTANT(CR,LF,LF,
                              'The new destination address is ',0),

                    STAT_ADDR(*) BYTE CONSTANT(CR,LF,LF,
                              'Enter the station address: __',BS,BS,0),

                    S_ADDR_ACK(*) BYTE CONSTANT(CR,LF,LF,
                              'The new station address is ',0),

                    ADDR_ACK_FIN(*) BYTE CONSTANT('H',CR,LF,LF,0),



                    SIGN_ON(*)  BYTE CONSTANT(CR, LF,LF,
                    '(\/)  RUPI-44 Secondary Station', CR, LF,
                    ' \/', CR, LF, LF,
                    '1 - Set the Station Address',LF,CR,
                    '2 - Set the Destination Address',CR,LF,
                    '3 - Go Online',CR,LF,
                    '4 - Go Offline',CR,LF,
                    '5 - Return to terminal mode',CR,LF,LF,
                    '    Enter option: _', BS, 0),

                    FIN(*) BYTE CONSTANT(CR,LF,LF,0),


                    /* Characters Received From the Terminal */

                    HEX_TABLE(17) BYTE CONSTANT('0123456789ABCDEF',BEL),
                    MENU_CHAR(6) BYTE CONSTANT('12345',BEL),

                      /* Flags and Bits */

                    XMIT_BUFFER_EMPTY       BIT     EXTERNAL, /* Semaphore  for RUPI SIOU Transmit Buffer */
                    STOP_BIT                BIT     AT(147) REG, /* Terminal parameters */
                    ECHO                    BIT     AT(084H) REG,
                    WAIT                    BIT,            /* Timeout Flag */
                    ERROR_FLAG              BIT,            /* Error message Flag */

                      /* Peripheral Addresses */

                    USART_STATUS            BYTE    AT(0801H) AUXILIARY,
                    USART_DATA              BYTE    AT(0800H) AUXILIARY,
                    TIMER_CONTROL           BYTE    AT(1003H) AUXILIARY,
                    TIMER_0                 BYTE    AT(1000H) AUXILIARY,
                    TIMER_1                 BYTE    AT(1001H) AUXILIARY,
                    TIMER_2                 BYTE    AT(1002H) AUXILIARY;


                        /* External Procedures */
```

296166-52

```
PL/M-51 COMPILER    Application Module: Async/SDLC Protocol converter    18:50:53  09/19/83  PAGE   3


   6   2     POWER_ON_D: PROCEDURE   EXTERNAL;
   7   1     END POWER_ON_D;

   8   2     CLOSE: PROCEDURE EXTERNAL USING 2;
   9   1     END CLOSE;

  10   2     OPEN: PROCEDURE EXTERNAL USING 2;
  11   1     END OPEN;

  12   2     TRANSMIT:   PROCEDURE   (XMIT_BUFFER_LENGTH)   BYTE   EXTERNAL;
  13   2                 DECLARE XMIT_BUFFER_LENGTH  BYTE;
  14   1     END TRANSMIT;

                        /* Local Procedures */

  15   2     TIMER_0_INT: PROCEDURE INTERRUPT 1 USING 1;
  16   2         WAIT=0;
  17   1     END TIMER_0_INT;

  18   2     POWER_ON:   PROCEDURE USING 0;

  19   2             DECLARE TEMP    BYTE    AUXILIARY;

  20   2             SMD=54H;              /* Using DPLL, NRZI, PFS, TIMER 1, @ 62.5 Kbps */
  21   2             TMOD=21H;             /* Timer 0 16 bit, Timer 1 auto reload */
  22   2             TH1=0FFH;
  23   2             TCON=40H;

  24   2             TIMER_CONTROL=37H;   /* Initialize USART's system clock; 8254 */
  25   2             TIMER_0=04H;
  26   2             TIMER_0=00H;
  27   2             TIMER_CONTROL=77H;  /* Initialize TxC, RxC  */

                /*      Definition for dip switch tied to P1.0 to P1.6

                  Bit Rate    3       2       1
                  ------------------------------
                    300       on      on      on
                   1200       on      on      off
                   2400       on      off     on
                   4800       on      off     off
                   9600       off     on      on
                  19200       off     on      off

                  Stop bit     4
                  ----------------
                      1        on
                      2        off

                  Parity       6       5
                  ------------------------
                  off          on      on
                  odd          on      off
                  off          off     on
                  even         off     off
```

296166-53

```
PL/M-51 COMPILER    Application Module: Async/SDLC Protocol converter          18:50:53  09/19/83  PAGE   4


                        Echo        7
                        ____________
                        on          on
                        off         off         */

  28   2              TEMP=P1 AND 07H;  /* Read the dip switch to determine the bit rate */
  29   2              IF TEMP>5
                        THEN TEMP=0;
  31   3              DO CASE TEMP;
                /* 300 */
  32   4                 DO;
  33   4                    TIMER_1=83H;
  34   4                    TIMER_1=20H;
  35   4                 END;

  36   4            /* 1200 */  DO;
  37   4                    TIMER_1=20H;
  38   4                    TIMER_1=05H;
  39   4                 END;

  40   4            /* 2400 */  DO;
  41   4                    TIMER_1=60H;
  42   4                    TIMER_1=02H;
  43   4                 END;

  44   4            /* 4800 */  DO;
  45   4                    TIMER_1=30H;
  46   4                    TIMER_1=01H;
  47   4                 END;

  48   4            /* 9600 */  DO;
  49   4                    TIMER_1=65H;
  50   4                    TIMER_1=0;
  51   4                 END;

  52   4            /* 19200 */ DO;
  53   4                    TIMER_1=33H;
  54   4                    TIMER_1=0;
  55   4                 END;
  56   3              END;

  57   2              USART_STATUS=0;   /* Software power-on reset for 8251A */
  58   2              USART_STATUS=0;
  59   2              USART_STATUS=0;
  60   2              USART_STATUS=40H;

  61   2              TEMP=0AH;         /* Determine the parity and # of stop bits */
  62   2              TEMP=TEMP OR (P1 AND 30H);
  63   2              IF STOP_BIT=1
                        THEN TEMP=TEMP OR 0C0H;
  65   2              ELSE TEMP=TEMP OR 40H;

  66   2              USART_STATUS=TEMP;  /* USART Mode Word */
  67   2              USART_STATUS, USART_CMD=27H; /*USART Command Word RTS, RxE, DTR, TxEN=1*/

  68   2              STAD=0FFH;
```

296166-54

```
PL/M-51 COMPILER    Application Module: Async/SDLC Protocol converter          18:50:53  09/19/83  PAGE   5


  69   2              SEND_DATA=0; /* Intialize Flags */
  70   2             IN_PTR_T, OUT_PTR_T, IN_PTR_R, OUT_PTR_R = 0;/*Initialize FIFO PTRs*/
  71   2              BUFFER_STATUS_T, BUFFER_STATUS_R= EMPTY;
  72   2              CALL POWER_ON_D;
  73   2              IP=01H;           /* USART's RxRdy is the highest priority */
                                          /* Both external interrupts are level triggered*/
  74   2              IE=93H;           /* Enable USART RxRdy, SI, and Timer 0 interrupts*/
  75   2              ERROR_FLAG=0;
  76   1      END POWER_ON;
  77   2      FIFO_R_IN: PROCEDURE (CHAR) USING 1;
  78   2          DECLARE CHAR    BYTE;
  79   2          FIFO_R(IN_PTR_R)=CHAR;
  80   2          IN_PTR_R=IN_PTR_R+1;
  81   2          IF BUFFER_STATUS_R=EMPTY
                  THEN DO;
  83   3                   EA=0;
  84   3                   BUFFER_STATUS_R=INUSE;
  85   3                   EX1=1;       /* Enable USART's TxD interrupt */
  86   3                   EA=1;
  87   3               END;
  88   2          ELSE IF ((BUFFER_STATUS_R=INUSE) AND (IN_PTR_R=OUT_PTR_R))
                       THEN BUFFER_STATUS_R=FULL;
  90   1      END FIFO_R_IN;
  91   2      FIFO_R_OUT: PROCEDURE BYTE USING 1;
  92   2          DECLARE CHAR    BYTE    AUXILIARY;
  93   2          CHAR=FIFO_R(OUT_PTR_R);
  94   2          OUT_PTR_R=OUT_PTR_R+1;
  95   2          IF OUT_PTR_R=IN_PTR_R
                  THEN DO;
  97   3                   EX1=0;   /* Shut off TxD interrupt */
  98   3                   BUFFER_STATUS_R=EMPTY;
  99   3               END;
 100   2          ELSE IF ((BUFFER_STATUS_R=FULL) AND (OUT_PTR_R-20=IN_PTR_R))
                       THEN BUFFER_STATUS_R=INUSE;
 102   2          RETURN CHAR;
 103   1      END FIFO_R_OUT;
 104   2      USART_XMIT_INT: PROCEDURE INTERRUPT 2 USING 1;
```

296166-55

```
PL/M-51 COMPILER    Application Module: Async/SDLC Protocol converter          18:50:53  09/19/83  PAGE   6


 105   2          DECLARE
                        MESSAGE BASED ERR_MESSAGE_PTR(1)     BYTE    CONSTANT;

 106   2          IF ERROR_FLAG
                   THEN DO;
 108   3                    IF MESSAGE(ERR_MESSAGE_INDEX)<>0  /* Then continue to send the message */
                              THEN DO;
 110   4                              USART_DATA = MESSAGE(ERR_MESSAGE_INDEX);
 111   4                              ERR_MESSAGE_INDEX=ERR_MESSAGE_INDEX+1;
 112   4                           END;

 113   4                    ELSE DO; /* If message is done reset ERROR_FLAG and shut off interrupt if FIFO is empty */
 114   4                           ERROR_FLAG=0;
 115   4                           IF BUFFER_STATUS_R = EMPTY
                                    THEN EX1=0;
 117   4                        END;
 118   3                 END;

 119   2          ELSE USART_DATA=FIFO_R_OUT;

 120   1      END USART_XMIT_INT;


 121   2      SIU_RECV: PROCEDURE (LENGTH) PUBLIC USING 1;

 122   2          DECLARE LENGTH  BYTE,
                         I       BYTE    AUXILIARY;

 123   3          DO I=0 TO LENGTH-1;
 124   4              DO WHILE BUFFER_STATUS_R=FULL; /* Check to see if fifo is full */
 125   4              END;
 126   3              CALL FIFO_R_IN(SIU_RECV_BUFFER(I));
 127   3          END;

 128   1      END SIU_RECV;


 129   2      FIFO_T_IN: PROCEDURE (CHAR) USING 2;

 130   2          DECLARE CHAR        BYTE;

 131   2          FIFO_T(IN_PTR_T)=CHAR;
 132   2          IN_PTR_T=IN_PTR_T+1;
 133   2          IF CHAR=LF
                   THEN SEND_DATA=SEND_DATA+1;

 135   2          IF BUFFER_STATUS_T=EMPTY
                   THEN BUFFER_STATUS_T=INUSE;
 137   2          ELSE IF ((BUFFER_STATUS_T=INUSE) AND (IN_PTR_T+20=OUT_PTR_T))
                        THEN DO; /* Stop reception using CTS */
 139   3                          USART_STATUS, USART_CMD=USART_CMD AND NOT(20H);
 140   3                          BUFFER_STATUS_T=FULL;
 141   3                          IF SEND_DATA=0
                                    THEN SEND_DATA=1;/*If the buffer is full and no LF
                                                       has been received then send data */
 143   3                     END;
 144   1      END FIFO_T_IN;
```

296166-56

```
 145   2      FIFO_T_OUT: PROCEDURE BYTE ;

 146   2          DECLARE CHAR    BYTE    AUXILIARY;

 147   2          CHAR=FIFO_T(OUT_PTR_T);
 148   2          OUT_PTR_T=OUT_PTR_T+1;
 149   2          IF OUT_PTR_T=IN_PTR_T   /* Then FIFO_T is empty */
                     THEN DO;
 151   3                  EA=0;
 152   3                  BUFFER_STATUS_T=EMPTY;
 153   3                  SEND_DATA=0;
 154   3                  EA=1;
 155   3                END;
 156   2          ELSE IF ((BUFFER_STATUS_T=FULL) AND (OUT_PTR_T-80=IN_PTR_T))
                     THEN DO;
 158   3                  USART_STATUS, USART_CMD=USART_CMD OR 20H;
 159   3                  BUFFER_STATUS_T=INUSE;
 160   3                END;
 161   2          IF (CHAR=LF AND SEND_DATA>0) THEN SEND_DATA=SEND_DATA-1;
 163   2          RETURN CHAR;
 164   1      END FIFO_T_OUT;

 165   2      ERROR: PROCEDURE (STATUS) USING 2;

 166   2          DECLARE STATUS      BYTE;

 167   2          IF (STATUS AND 08H)<>0
                     THEN ERR_MESSAGE_PTR=.PARITY;
 169   2          ELSE IF (STATUS AND 10H)<>0
                     THEN ERR_MESSAGE_PTR=.OVER_RUN;
 171   2          ELSE IF (STATUS AND 20H)<>0
                     THEN ERR_MESSAGE_PTR=.FRAME;

 173   2          USART_STATUS=(USART_CMD OR 10H); /* Reset error flags on USART */

 174   2          ERR_MESSAGE_INDEX = 0;
 175   2          ERROR_FLAG=1;
 176   2          EX1=1;          /* Turn on Tx Interrupt */

 177   1      END ERROR;

 178   2      LINK_DISC: PROCEDURE ;

              /* This procedure sends the message 'Unable to Get Online' to the terminal */

 179   2          DECLARE MESSAGE_PTR WORD    AUXILIARY,
                         MESSAGE     BASED   MESSAGE_PTR(1)      BYTE    CONSTANT,
                         J           BYTE    AUXILIARY,
                         EX1_STORE   BIT;


 180   2          EX1_STORE=EX1;    /* Shut off async transmit interrupt */
 181   2          EX1=0;
 182   2          MESSAGE_PTR=.LINK;
 183   2          J=0;
 184   3          DO WHILE (MESSAGE(J)<>0);
```

296166-57

```
PL/M-51 COMPILER    Application Module: Async/SDLC Protocol converter          18:50:53  09/19/83  PAGE   8


  185   4              DO WHILE (USART_STATUS AND 01H)=0;  /* Wait for TxRDY on USART */
  186   4              END;
  187   3              USART_DATA=MESSAGE(J);
  188   3              J=J+1;
  189   3          END;
  190   2          EX1=EX1_STORE;    /* Restore async transmit interrupt */
  191   1      END LINK_DISC;


  192   2      CO: PROCEDURE (CHAR) USING 2;
  193   2              DECLARE CHAR    BYTE;

  194   3              DO WHILE (USART_STATUS AND 01H) = 0;
  195   3              END;
  196   2              USART_DATA=CHAR;

  197   1      END CO;

  198   2      CI: PROCEDURE BYTE USING 2;

  199   3          DO WHILE (USART_STATUS AND 02H) = 0;
  200   3          END;
  201   2          RETURN USART_DATA;

  202   1      END CI;


  203   2      GET_HEX: PROCEDURE BYTE USING 2;

  204   2          DECLARE CHAR    BYTE    AUXILIARY,
                          I       BYTE    AUXILIARY;

  205   2          LO: CHAR=CI;

  206   3              DO I=0 TO 15;
  207   3                  IF CHAR=HEX_TABLE(I)
                              THEN GOTO L1;
  209   3              END;

  210   2          L1: CALL CO(HEX_TABLE(I));
  211   2              IF I=16
                          THEN GOTO LO;

  213   2              RETURN I;

  214   1      END GET_HEX;

  215   2      OUTPUT_MESSAGE: PROCEDURE (MESSAGE_PTR) USING 2;
  216   2          DECLARE MESSAGE_PTR WORD,
                          MESSAGE     BASED   MESSAGE_PTR(1)  BYTE CONSTANT,
                          I           BYTE    AUXILIARY;

  217   2          I=0;

  218   3          DO WHILE MESSAGE(I) <> 0;
  219   3              CALL CO(MESSAGE(I));
  220   3              I=I+1;
```

296166-58

```
PL/M-51 COMPILER   Application Module: Async/SDLC Protocol converter          18:50:53  09/19/83  PAGE   9

  221   3          END;
  222   1      END OUTPUT_MESSAGE;


  223   2      MENU: PROCEDURE USING 2;

  224   2          DECLARE I               BYTE    AUXILIARY,
                          CHAR            BYTE    AUXILIARY,
                          STATION_ADDRESS BYTE    AUXILIARY;

  225   2      START:
                  CALL OUTPUT_MESSAGE(.SIGN_ON);

  226   2          M0: CHAR=CI;     /* Read a character */

  227   3              DO I=0 TO 4;
  228   3                  IF CHAR=MENU_CHAR(I)
                              THEN GOTO M1;
  230   3              END;

  231   2          M1: CALL CO(MENU_CHAR(I));
  232   2              IF I=5
                          THEN GOTO M0;

  234   3          DO CASE I;

  235   4              DO;
  236   4                  CALL OUTPUT_MESSAGE(.STAT_ADDR);

  237   4                  STATION_ADDRESS=SHL(GET_HEX,4);

  238   4                  STATION_ADDRESS=(STATION_ADDRESS OR GET_HEX);

  239   4                  STAD=STATION_ADDRESS;

  240   4                  CALL OUTPUT_MESSAGE(.S_ADDR_ACK);

  241   4                  CALL CO(HEX_TABLE(SHR(STATION_ADDRESS,4)));
  242   4                  CALL CO(HEX_TABLE(0FH AND STATION_ADDRESS));

  243   4                  CALL OUTPUT_MESSAGE(.ADDR_ACK_FIN);
  244   4              END;

  245   4              DO;

  246   4                  CALL OUTPUT_MESSAGE(.DEST_ADDR);

  247   4                  DESTINATION_ADDRESS=SHL(GET_HEX,4);

  248   4                  DESTINATION_ADDRESS=(DESTINATION_ADDRESS OR GET_HEX );

  249   4                  CALL OUTPUT_MESSAGE(.D_ADDR_ACK);
```

296166-59

```
PL/M-51 COMPILER    Application Module: Async/SDLC Protocol converter                18:50:53  09/19/83  PAGE  10


 250   4                  CALL CO(HEX_TABLE(SHR(DESTINATION_ADDRESS,4)));
 251   4                  CALL CO(HEX_TABLE(OFH AND DESTINATION_ADDRESS));

 252   4                  CALL OUTPUT_MESSAGE(.ADDR_ACK_FIN);
 253   4              END;

 254   4              DO;
 255   4                  CALL OUTPUT_MESSAGE(.FIN);
 256   4                  CALL OPEN;
 257   4              END;


 258   4              DO;
 259   4                  CALL OUTPUT_MESSAGE(.FIN);
 260   4                  CALL CLOSE;
 261   4              END;

 262   3              CALL OUTPUT_MESSAGE(.FIN);

 263   3          END; /* DO CASE */

 264   1      END MENU;


 265   2      USART_RECV_INT: PROCEDURE INTERRUPT 0 USING 2;

 266   2          DECLARE CHAR        BYTE    AUXILIARY,
                         STATUS      BYTE    AUXILIARY;


 267   2          CHAR=USART_DATA;
 268   2          STATUS=USART_STATUS AND 38H;
 269   2          IF STATUS<>0
                     THEN CALL ERROR(STATUS);
 271   2          ELSE IF CHAR=ESC
                     THEN CALL MENU;
 273   3          ELSE DO;
 274   3                  CALL FIFO_T_IN(CHAR);
 275   3                  IF ECHO=0
                             THEN CALL CO(CHAR);
 277   3               END;

 278   1      END USART_RECV_INT;


 279   1      BEGIN:
              CALL POWER_ON;

 280   2          DO FOREVER;
 281   2              IF SEND_DATA>0
                         THEN DO;
 283   4                           DO WHILE NOT(XMIT_BUFFER_EMPTY); /*Wait until SIU_XMIT_BUFFer
                                                                      is empty */
 284   4                           END;
 285   3                           LENGTH, CHAR =1;
 286   3                           SIU_XMIT_BUFFER(0)=DESTINATION_ADDRESS;
 287   4                           DO WHILE ((CHAR<>LF) AND (LENGTH<BUFFER_LENGTH) AND (BUFFER_STATUS_T<>EMPTY));
```

296166-60

```
PL/M-51 COMPILER    Application Module: Async/SDLC Protocol converter                18:50:53  09/19/83  PAGE  11


  288   4                           CHAR=FIFO_T_OUT;
  289   4                           SIU_XMIT_BUFFER(LENGTH)=CHAR;
  290   4                           LENGTH=LENGTH+1;
  291   4                       END;

            /* If the line entered at the terminal is greater than BUFFER_LENGTH char, send the
               first BUFFER_LENGTH char, then send the rest; since the SIU buffer is only BUFFER_LENGTH bytes */
  292   3       L1:                 I=0; /* Use I to count the number of unsuccesful
                                          transmits */

  293   3       RETRY:          RESULT=TRANSMIT(LENGTH); /* Send the message */
  294   3                           IF RESULT<>DATA_TRANSMITTED
                                      THEN DO;
                                /* Wait 50 msec for link to connect then try again */
  296   4                               WAIT=1;
  297   4                               TH0=3CH;
  298   4                               TL0=0AFH;
  299   4                               TR0=1;
  300   5                               DO WHILE WAIT;
  301   5                               END;
  302   4                               TR0=0;
  303   4                               I=I+1;
  304   5                               IF I>100 THEN DO;  /* Wait 5 sec to get on line else
                                                             send error message to terminal
                                                             and try again  */
  306   5                                                      CALL LINK_DISC;
  307   5                                                      GOTO L1;
  308   5                                                  END;
  309   4                               GOTO RETRY;
  310   4                           END;
  311   3                   END;

  312   2           END;
  313   1       END USER$MOD;



WARNINGS:
    2 IS THE HIGHEST USED INTERRUPT


MODULE INFORMATION:                 (STATIC+OVERLAYABLE)
    CODE SIZE                       = 06B2H      1714D
    CONSTANT SIZE                   = 01CFH       463D
    DIRECT VARIABLE SIZE            =   00H+05H     0D+  5D
    INDIRECT VARIABLE SIZE          =   00H+00H     0D+  0D
    BIT SIZE                        =   02H+01H     2D+  1D
    BIT-ADDRESSABLE SIZE            =   00H+00H     0D+  0D
    AUXILIARY VARIABLE SIZE         = 021FH       543D
    MAXIMUM STACK SIZE              = 0028H        40D
    REGISTER-BANK(S) USED:            0 1 2
    713 LINES READ
    0 PROGRAM ERROR(S)
END OF PL/M-51 COMPILATION
```

296166-61

```
PL/M-51 COMPILER    RUPI-44 Primary Station                    20:47:13  09/26/83  PAGE   1


ISIS-II PL/M-51 V1.0
COMPILER INVOKED BY:   :F2:PLM51 :F2:PNOTE.SRC


                $TITLE      ('RUPI-44 Primary Station')
                $DEBUG
                $REGISTERBANK(0)
  1   1         MAIN$MOD:DO;

                /* To save paper the RUPI registers are not listed, but this is the statement
                   used to include them:  $INCLUDE (:F2:REG44.DCL) */

                $NOLIST

  5   1         DECLARE LIT             LITERALLY   'LITERALLY',
                        TRUE            LIT         '0FFH',
                        FALSE           LIT         '00H',
                        FOREVER         LIT         'WHILE 1';

                /* SDLC COMMANDS AND RESPONSES */

  6   1         DECLARE SNRM            LIT         '93H',
                        UA              LIT         '73H',
                        DISC            LIT         '53H',
                        DM              LIT         '1FH',
                        FRMR            LIT         '97H',
                        REQ_DISC        LIT         '53H',
                        UP              LIT         '33H',
                        TEST            LIT         '0F3H',
                        RR              LIT         '11H',
                        RNR             LIT         '15H',


                              /* REMOTE STATION BUFFER STATUS */

                        BUFFER_READY        LIT         '0',
                        BUFFER_NOT_READY    LIT         '1',

                              /* STATION STATES */
                        DISCONNECT_S    LIT         '00H', /* LOGICALLY DISCONNECTED STATE*/
                        GO_TO_DISC      LIT         '01H',
                        I_T_S           LIT         '02H', /* INFORMATION TRANSFER STATE */


                        /* PARAMETERS PASSED TO XMIT_I_T_S */
                        T_I_FRAME           LIT     '00H',
                        T_RR                LIT     '01H',
                        T_RNR               LIT     '02H',


                        /* SECONDARY STATION IDENTIFICATION */

                        NUMBER_OF_STATIONS  LIT     '2',
                        SECONDARY_ADDRESSES(NUMBER_OF_STATIONS)
                                            BYTE    CONSTANT(55H,43H),
```

296166-62

```
PL/M-51 COMPILER    RUPI-44 Primary Station                          20:47:13  09/26/83  PAGE   2


                            /* Remote Station Database */

                        RSD(NUMBER_OF_STATIONS) STRUCTURE
                        (STATION_ADDRESS    BYTE,
                         STATION_STATE      BYTE,
                         NS                 BYTE,
                         NR                 BYTE,
                         BUFFER_STATUS      BYTE,  /* The status of the secondary stations buffer */
                         INFO_LENGTH        BYTE,
                         DATA(64)           BYTE)  AUXILIARY,

                            /* VARIABLES */
                        STATION_NUMBER      BYTE   AUXILIARY,
                        RECV_FIELD_LENGTH   BYTE   AUXILIARY,
                        WAIT                BIT,

                            /* BUFFERS */
                        SIU_XMIT_BUFFER(64)     BYTE    IDATA,
                        SIU_RECV_BUFFER(64)     BYTE;


  7   2     POWER_ON: PROCEDURE ;

  8   2         DECLARE I   BYTE    AUXILIARY;

  9   2             TBS=.SIU_XMIT_BUFFER(0);
 10   2             RBS=.SIU_RECV_BUFFER(0);
 11   2             RBL=64;       /* 64 Byte receive buffer */
 12   2             RBE=1;        /* Enable the SIU's receiver */

 13   3             DO I= 0 TO NUMBER_OF_STATIONS-1;

 14   3                 RSD(I).STATION_ADDRESS=SECONDARY_ADDRESSES(I);
 15   3                 RSD(I).STATION_STATE=DISCONNECT_S;
 16   3                 RSD(I).BUFFER_STATUS=BUFFER_NOT_READY;
 17   3                 RSD(I).INFO_LENGTH=0;

 18   3             END;

 19   2             SMD=54H; /* Using DPLL, NRZI, PFS, TIMER 1, @ 62.5 Kbps */
 20   2             TMOD=21H;
 21   2             TH1=0FFH;
 22   2             TCON=40H; /* Use timer 0 for receive time out interrupt */
 23   2             IE=82H;

 24   1     END POWER_ON;

 25   2     XMIT: PROCEDURE (CONTROL_BYTE);

 26   2         DECLARE CONTROL_BYTE    BYTE;

 27   2             TCB=CONTROL_BYTE;
 28   2             TBF=1;
```

296166-63

```
PL/M-51 COMPILER    RUPI-44 Primary Station                    20:47:13  09/26/83  PAGE   3

  29   2              RTS=1;
  30   3              DO WHILE NOT SI;
  31   3              END;
  32   2              SI=0;

  33   1      END XMIT;


  34   2      TIMER_0_INT: PROCEDURE INTERRUPT 1  USING 1;
  35   2          WAIT=0;
  36   1      END TIMER_0_INT;


  37   2      TIME_OUT: PROCEDURE BYTE;          /* Time_out returns true if there wasn't
                                                    a frame received within 200 msec.
                                                    If there was a frame received within
                                                    200 msec then time_out returns false. */
  38   2          DECLARE     I   BYTE    AUXILIARY;

  39   3          DO I=0 TO 3;

  40   3              WAIT=1;
  41   3              THO=3CH;
  42   3              TLO=0AFH;
  43   3              TRO=1;
  44   4                  DO WHILE WAIT;
  45   4                      IF SI=1
                                  THEN GOTO T_01;
  47   4                  END;
  48   3          END;

  49   2          RETURN TRUE;

  50   2      T_01:
                  SI=0;
  51   2              RETURN FALSE;

  52   1      END TIME_OUT;

  53   2      SEND_DISC: PROCEDURE;

  54   2                  TBL=0;
  55   2                  CALL XMIT(DISC);
  56   2                  IF TIME_OUT=FALSE
                              THEN IF RCB=UA OR RCB=DM
  57   3                                  THEN DO;
  59   3                                      RSD(STATION_NUMBER).BUFFER_STATUS=BUFFER_NOT_READY;
  60   3                                      RSD(STATION_NUMBER).STATION_STATE=DISCONNECT_S;
  61   3                                  END;
  62   2                  RBE=1;

  63   1      END SEND_DISC;


  64   2      SEND_SNRM: PROCEDURE;
  65   2          TBL=0;
```

296166-64

```
  66   2          CALL XMIT(SNRM);
  67   2          IF (TIME_OUT=FALSE) AND (RCB=UA)
                         THEN DO;
  69   3                           RSD(STATION_NUMBER).STATION_STATE=I_T_S;
  70   3                           RSD(STATION_NUMBER).NS=0;
  71   3                           RSD(STATION_NUMBER).NR=0;
  72   3                      END;
  73   2          RBE=1;

  74   1      END SEND_SNRM;


  75   2      CHECK_NS: PROCEDURE BYTE;

                /* Check the Ns Field of the received frame.  If Nr(P)=Ns(S) return true */

  76   2          IF (RSD(STATION_NUMBER).NR=(SHR(RCB,1) AND 07H))
                    THEN RETURN TRUE;
  78   2                 ELSE RETURN FALSE;

  79   1      END CHECK_NS;

  80   2      CHECK_NR: PROCEDURE BYTE;

                /* Check the Nr field of the received frame.  If Ns(P)+1=Nr(S) then the frame
                   has been acknowledged, else if Ns(P)=Nr(S) then the frame has not been
                   acknowledged, else reset the secondary */

  81   2          IF (((RSD(STATION_NUMBER).NS + 1) AND 07H) = SHR(RCB,5))
                    THEN DO;
  83   3                      RSD(STATION_NUMBER).NS=((RSD(STATION_NUMBER).NS+1) AND 07H);
  84   3                      RSD(STATION_NUMBER).INFO_LENGTH=0;
  85   3                 END;
  86   2          ELSE IF (RSD(STATION_NUMBER).NS <> SHR(RCB,5))
                         THEN RETURN FALSE;

  88   2          RETURN TRUE;

  89   1      END CHECK_NR;


  90   2      RECEIVE: PROCEDURE ;

  91   2            DECLARE     I   BYTE    AUXILIARY;

  92   2            RSD(STATION_NUMBER).BUFFER_STATUS=BUFFER_READY;

                /* If an RNR was received buffer_status will be changed in the supervisory
                   frame decode section futher down in this procedure, any other response
                   means the remote stations buffer is ready  */

  93   2            IF (RCB AND 01H)=0
                    THEN DO;  /* I Frame Received */
  95   3                         IF (CHECK_NS=TRUE AND BOV=0 AND CHECK_NR=TRUE)
                                    THEN DO;
  97   4                                    RSD(STATION_NUMBER).NR=((RSD(STATION_NUMBER).NR+1) AND 07H);
  98   4                                    RBP=1;
```

296166-65

```
 99   4                                    RECV_FIELD_LENGTH=RFL-1;
100   4                                 END;

101   3                              ELSE RSD(STATION_NUMBER).STATION_STATE=GO_TO_DISC;
102   3                           END;
103   2                  ELSE IF (RCB AND 03H)=01H
                              THEN DO;  /* Supervisory frame received */
105   3                                 IF CHECK_NR=FALSE
                                           THEN RSD(STATION_NUMBER).STATION_STATE=GO_TO_DISC;

107   3                                    ELSE IF ((RCB AND 0FH)=05H) /* then RNR */
                                              THEN RSD(STATION_NUMBER).BUFFER_STATUS=BUFFER_NOT_READY;
109   3                                 END;

110   3                  ELSE DO;          /* Unnumbered frame or unknown frame received */
111   3                           IF RCB=FRMR
                                  THEN DO;    /* If FRMR was received check Nr for an
                                                 acknowledged I frame  */
113   4                                    RCB=SIU_RECV_BUFFER(1);
114   4                                    I=CHECK_NR;
115   4                                 END;
116   3                           RSD(STATION_NUMBER).STATION_STATE=GO_TO_DISC;
117   3                        END;

118   2                  RBE=1;

119   1      END RECEIVE;



120   2      XMIT_I_T_S: PROCEDURE (TEMP);

121   2          DECLARE     TEMP    BYTE;

122   2              IF TEMP=T_I_FRAME
                      THEN DO;    /* Transmit I frame */
                              /* Transfer the station buffer into internal ram */

124   4                         DO TEMP=0 TO RSD(STATION_NUMBER).INFO_LENGTH-1;
125   4                         SIU_XMIT_BUFFER(TEMP)=RSD(STATION_NUMBER).DATA(TEMP);
126   4                         END;

                              /* Build the I frame control field */

127   3                         TEMP=(SHL(RSD(STATION_NUMBER).NR,5) OR SHL(RSD(STATION_NUMBER).NS,1) OR 10H);
128   3                         TBL=RSD(STATION_NUMBER).INFO_LENGTH;
129   3                         CALL XMIT(TEMP);
130   3                         IF TIME_OUT=FALSE
                                  THEN CALL RECEIVE;
132   3                      END;

133   3                  ELSE DO;   /* Transmit RR or RNR*/
134   3                       IF TEMP=T_RR
                                  THEN TEMP=RR;
136   3                       ELSE TEMP=RNR;
```

296166-66

```
PL/M-51 COMPILER    RUPI-44 Primary Station                          20:47:13  09/26/83  PAGE   6

 137   3                    TEMP=(SHL(RSD(STATION_NUMBER).NR,5) OR TEMP);
 138   3                    TBL=0;
 139   3                    CALL XMIT(TEMP);
 140   3                    IF TIME_OUT=FALSE
                               THEN CALL RECEIVE;
 142   3                END;
 143   1      END XMIT_I_T_S;

 144   2      BUFFER_TRANSFER: PROCEDURE;

 145   2          DECLARE     I       BYTE    AUXILIARY,
                               J       BYTE    AUXILIARY;


 146   3          DO I=0 TO NUMBER_OF_STATIONS-1;
 147   3              IF RSD(I).STATION_ADDRESS=SIU_RECV_BUFFER(0)
                         THEN GOTO T1;
 149   3          END;

 150   2        T1:   IF I=NUMBER_OF_STATIONS  /* If the addressed station does not exits,
                                                  then discard the data */
                         THEN DO;
 152   3                          RBP=0;
 153   3                          RETURN;
 154   3                      END;
 155   2                ELSE IF RSD(I).INFO_LENGTH=0
                             THEN DO;
 157   3                          RSD(I).INFO_LENGTH=RECV_FIELD_LENGTH;
 158   4                          DO J=1 TO RECV_FIELD_LENGTH;
 159   4                              RSD(I).DATA(J-1)=SIU_RECV_BUFFER(J);
 160   4                          END;
 161   3                          RBP=0;
 162   3                      END;

 163   1      END BUFFER_TRANSFER;


 164   1      BEGIN:
              CALL POWER_ON;

 165   2          DO FOREVER;

 166   3              DO STATION_NUMBER=0 TO NUMBER_OF_STATIONS-1;
 167   3                  STAD=RSD(STATION_NUMBER).STATION_ADDRESS;
 168   3                  IF RSD(STATION_NUMBER).STATION_STATE = DISCONNECT_S
                             THEN CALL SEND_SNRM;
 170   3                  ELSE IF RSD(STATION_NUMBER).STATION_STATE = GO_TO_DISC
                             THEN CALL SEND_DISC;
 172   3                  ELSE IF ((RSD(STATION_NUMBER).INFO_LENGTH>0) AND
                                  (RSD(STATION_NUMBER).BUFFER_STATUS=BUFFER_READY))
                             THEN CALL XMIT_I_T_S(T_I_FRAME);
 174   3                  ELSE IF RBP=0
                             THEN CALL XMIT_I_T_S(T_RR);
 176   3                  ELSE CALL XMIT_I_T_S(T_RNR);

 177   3                  IF RBP=1
                             THEN CALL BUFFER_TRANSFER;
```

296166-67

```
PL/M-51 COMPILER    RUPI-44 Primary Station                          20:47:13  09/26/83  PAGE   7

 179   3              END;

 180   2          END;

 181   1      END MAIN$MOD;



WARNINGS:
    1 IS THE HIGHEST USED INTERRUPT


MODULE INFORMATION:                 (STATIC+OVERLAYABLE)
    CODE SIZE                       = 053DH       1341D
    CONSTANT SIZE                   = 0002H          2D
    DIRECT VARIABLE SIZE            =   40H+02H    64D+  2D
    INDIRECT VARIABLE SIZE          =   40H+00H    64D+  0D
    BIT SIZE                        =   01H+00H     1D+  0D
    BIT-ADDRESSABLE SIZE            =   00H+00H     0D+  0D
    AUXILIARY VARIABLE SIZE         = 0093H       147D
    MAXIMUM STACK SIZE              = 0019H        25D
    REGISTER-BANK(S) USED:            0 1
    456 LINES READ
    0 PROGRAM ERROR(S)
END OF PL/M-51 COMPILATION
```

296166-68

RUPI™ Datasheet, Application Note, Article Reprint and Development Support Tools

15

Application Note, Article Reprint and Development Support Tools

PRELIMINARY

8044AH/8344AH/8744H
HIGH PERFORMANCE 8-BIT MICROCONTROLLER WITH ON-CHIP SERIAL COMMUNICATION CONTROLLER

- **8044AH—Includes Factory Mask Programmable ROM**
- **8344AH—For Use with External Program Memory**
- **8744H—Includes User Programmable/Eraseable EPROM**

8051 MICROCONTROLLER CORE

- **Optimized for Real Time Control 12 MHz Clock, Priority Interrupts, 32 Programmable I/O Lines, Two 16-bit Timer/Counters**
- **Boolean Processor**
- **4K × 8 ROM, 192 × 8 RAM**
- **64K Accessible External Program Memory**
- **64K Accessible External Data Memory**
- **4 μs Multiply and Divide**

SERIAL INTERFACE UNIT (SIU)

- **Serial Communication Processor that Operates Concurrently to CPU**
- **2.4 Mbps Maximum Data Rate**
- **375 Kbps using On-Chip Phase Locked Loop**
- **Communication Software in Silicon:**
  - **— Complete Data Link Functions**
  - **— Automatic Station Response**
- **Operates as an SDLC Primary or Secondary Station**

The RUPI-44 family integrates a high performance 8-bit Microcontroller, the Intel 8051 Core, with an Intelligent/high performance HDLC/SDLC serial communication controller, called the Serial Interface Unit (SIU). See Figure 1. This dual architecture allows complex control and high speed data communication functions to be realized cost effectively.

Specifically, the 8044's Microcontroller features: 4K byte On-Chip program memory space; 32 I/O lines; two 16-bit timer/event counters; a 5-source; 2-level interrupt structure; a full duplex serial channel; a Boolean processor; and on-chip oscillator and clock circuitry. Standard TTL and most byte-oriented MCS-80 and MCS-85 peripherals can be used for I/O amd memory expansion.

The Serial Interface Unit (SIU) manages the interface to a high speed serial link. The SIU offloads the On-Chip 8051 Microcontroller of communication tasks, thereby freeing the CPU to concentrate on real time control tasks.

The RUPI-44 family consists of the 8044, 8744, and 8344. All three devices are identical except in respect of on-chip program memory. The 8044 contains 4K bytes of mask-programmable ROM. User programmable EPROM replaces ROM in the 8744. The 8344 addresses all program memory externally.

The RUPI-44 devices are fabricated with Intel's reliable +5 volt, silicon-gate HMOSII technology and packaged in a 40-pin DIP.

The 8744H is available in a hermetically sealed, ceramic, 40-lead dual in-line package which includes a window that allows for EPROM erasure when exposed to ultraviolet light (See Erasure Characteristics). During normal operation, ambient light may adversely affect the functionality of the chip. Therefore applications which expose the 8744H to ambient light may require an opaque label over the window.

8044's Dual Controller Architecture

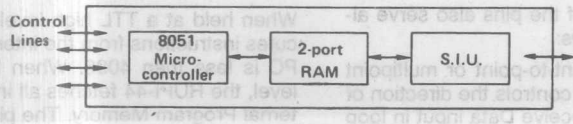


Figure 1. Dual Controller Architecture

October 1987
Order Number: 231663-004

Table 1. RUPI™-44 Family Pin Description

VSS

Circuit ground potential.

VCC

+5V power supply during operation and program verification.

PORT 0

Port 0 is an 8-bit open drain bidirectional I/O port. It is also the multiplexed low-order address and data bus when using external memory. It is used for data output during program verification. Port 0 can sink/source eight LS TTL loads (six in 8744).

PORT 1

Port 1 is an 8-bit quasi-bidirectional I/O port. It is used for the low-order address byte during program verification. Port 1 can sink/source four LS TTL loads.

In non-loop mode two of the I/O lines serve alternate functions:

— $\overline{RTS}$ (P1.6). Request-to-Send output. A low indicates that the RUPI-44 is ready to transmit.

— $\overline{CTS}$ (P1.7) Clear-to-Send input. A low indicates that a receiving station is ready to receive.

PORT 2

Port 2 is an 8-bit quasi-bidirection I/O port. It also emits the high-order address byte when accessing external memory. It is used for the high-order address and the control signals during program verification. Port 2 can sink/source four LS TTL loads.

PORT 3

Port 3 is an 8-bit quasi-bidirectional I/O port. It also contains the interrupt, timer, serial port and RD and WR pins that are used by various options. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. Port 3 can sink/source four LS LTT loads.

In addition to I/O, some of the pins also serve alternate functions as follows:

— I/$\overline{O}$ RxD (P3.0). In point-to-point or multipoint configurations, this pin controls the direction of pin P3.1. Serves as Receive Data input in loop and diagnostic modes.

— DATA TxD (P3.1) In point-to-point or multipoint configurations, this pin functions as data input/output. In loop mode, it serves as transmit pin. A '0' written to this pin enables diagnostic mode.

— $\overline{INT0}$ (P3.2). Interrupt 0 input or gate control input for counter 0.

— $\overline{INT1}$ (P3.3). Interrupt 1 input or gate control input for counter 1.

— TO (P3.4). Input to counter 0.

— SCLK T1 (P3.5). In addition to I/O, this pin provides input to counter 1 or serves as SCLK (serial clock) input.

— $\overline{WR}$ (P3.6). The write control signal latches the data byte from Port 0 into the External Data Memory.

— $\overline{RD}$ (P3.7). The read control signal enables External Data Memory to Port 0.

RST

A high on this pin for two machine cycles while the oscillator is running resets the device. A small external pulldown resistor ($\approx$8.2KΩ) from RST to V_{SS} permits power-on reset when a capacitor ($\approx$10μf) is also connected from this pin to V_{CC}.

ALE/$\overline{PROG}$

Provides Address Latch Enable output used for latching the address into external memory during normal operation. It is activated every six oscillator periods except during an external data memory access. It also receives the program pulse input for programming the EPROM version.

$\overline{PSEN}$

The Program Store Enable output is a control signal that enables the external Program Memory to the bus during external fetch operations. It is activated every six oscillator periods, except during external data memory accesses. Remains high during internal program execution.

$\overline{EA}$/VPP

When held at a TTL high level, the RUPI-44 executes instructions from the internal ROM when the PC is less than 4096. When held at a TTL low level, the RUPI-44 fetches all instructions from external Program Memory. The pin also receives the 21V EPROM programming supply voltage on the 8744.

Table 1. RUPI™-44 Family Pin Description (Continued)

XTAL 1

Input to the oscillator's high gain amplifier. Required when a crystal is used. Connect to VSS when external source is used on XTAL 2.

XTAL 2

Output from the oscillator's amplifier. Input to the internal timing circuitry. A crystal or external source can be used.

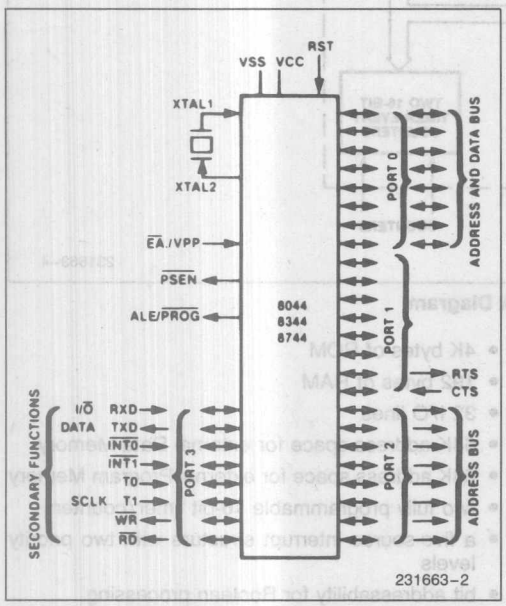


Figure 2. Logic Symbol

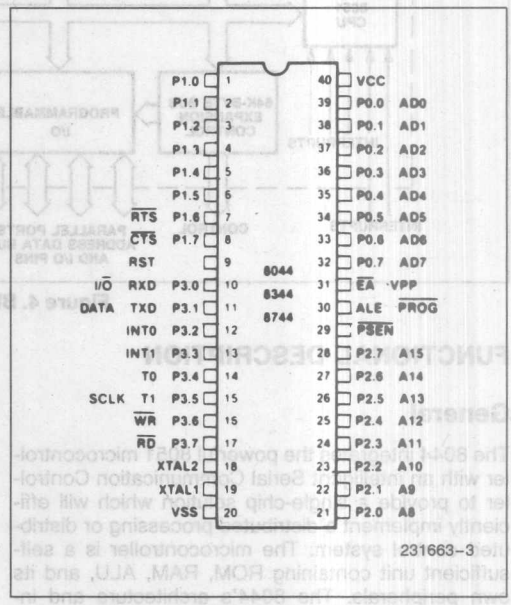


Figure 3A. DIP Pin Configuration

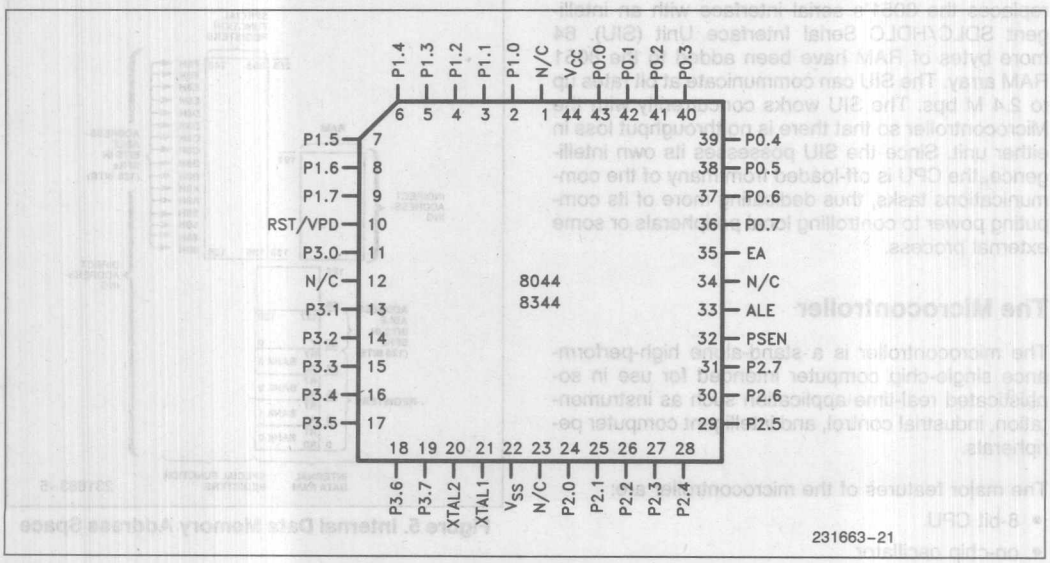


Figure 3B. PLCC Pin Configuration

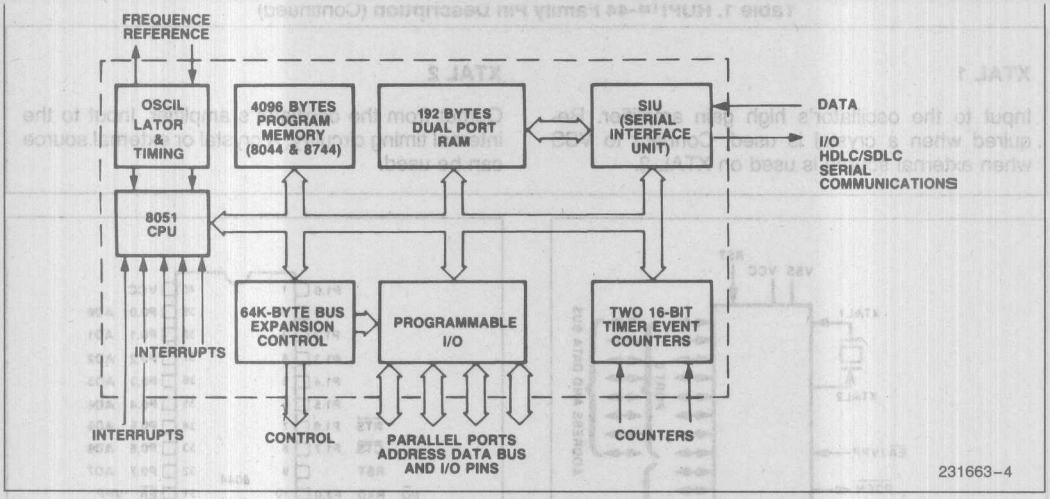


Figure 4. Block Diagram

FUNCTIONAL DESCRIPTION

General

The 8044 integrates the powerful 8051 microcontroller with an intelligent Serial Communication Controller to provide a single-chip solution which will efficiently implement a distributed processing or distributed control system. The microcontroller is a self-sufficient unit containing ROM, RAM, ALU, and its own peripherals. The 8044's architecture and instruction set are identical to the 8051's. The 8044 replaces the 8051's serial interface with an intelligent SDLC/HDLC Serial Interface Unit (SIU). 64 more bytes of RAM have been added to the 8051 RAM array. The SIU can communicate at bit rates up to 2.4 M bps. The SIU works concurrently with the Microcontroller so that there is no throughput loss in either unit. Since the SIU possesses its own intelligence, the CPU is off-loaded from many of the communications tasks, thus dedicating more of its computing power to controlling local peripherals or some external process.

The Microcontroller

The microcontroller is a stand-alone high-performance single-chip computer intended for use in sophisticated real-time application such as instrumentation, industrial control, and intelligent computer peripherals.

The major features of the microcontroller are:

- 8-bit CPU
- on-chip oscillator
- 4K bytes of ROM
- 192 bytes of RAM
- 32 I/O lines
- 64K address space for external Data Memory
- 64K address space for external Program Memory
- two fully programmable 16-bit timer/counters
- a five-source interrupt structure with two priority levels
- bit addressability for Boolean processing

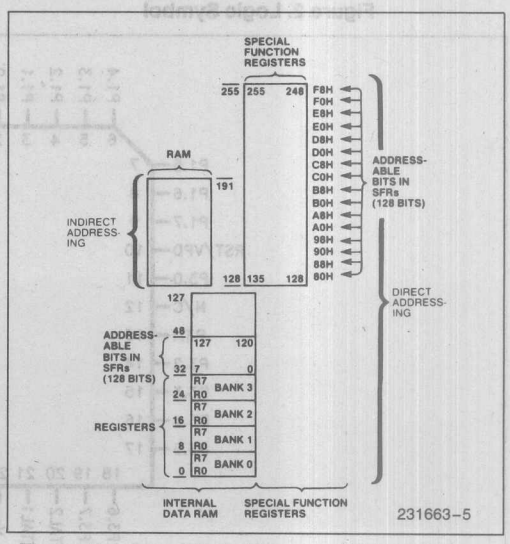


Figure 5. Internal Data Memory Address Space

- 1 μs instruction cycle time for 60% of the instructions 2 μs instruction cycle time for 40% of the instructions
- 4 μs cycle time for 8 by 8 bit unsigned Multiply/Divide

INTERNAL DATA MEMORY

Functionally the Internal Data Memory is the most flexible of the address spaces. The Internal Data Memory space is subdivided into a 256-byte Internal Data RAM address space and a 128-bit Special Function Register address space as shown in Figure 5.

The Internal Data RAM address space is 0 to 255. Four 8-Register Banks occupy locations 0 through 31. The stack can be located anywhere in the Internal Data RAM address space. In addition, 128 bit locations of the on-chip RAM are accessible through Direct Addressing. These bits reside in Internal Data RAM at byte locations 32 through 47. Currently locations 0 through 191 of the Internal Data RAM address space are filled with on-chip RAM.

Parallel I/O

The 8044 has 32 general-purpose I/O lines which are arranged into four groups of eight lines. Each group is called a port. Hence there are four ports; Port 0, Port 1, Port 2, and Port 3. Up to five lines from Port 3 are dedicated to supporting the serial channel when the SIU is invoked. Due to the nature of the serial port, two of Port 3's I/O lines (P3.0 and P3.1) do not have latched outputs. This is true whether or not the serial channel is used.

Port 0 and Port 2 also have an alternate dedicated function. When placed in the external access mode, Port 0 and Port 2 become the means by which the 8044 communicates with external program memory. Port 0 and Port 2 are also the means by which the 8044 communicates with external data memory. Peripherals can be memory mapped into the address space and controlled by the 8044.

Table 2. MCS®-51 Instruction Set Description

Mnemonic		Description	Byte	Cyc
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	1
ADD	A,direct	Add direct byte to Accumulator	2	1
ADD	A,@Ri	Add indirect RAM to Accumulator	1	1
ADD	A,#data	Add immediate data to Accumulator	2	1
ADDC	A,Rn	Add register to Accumulator with Carry	1	1
ADDC	A,direct	Add direct byte to A with Carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with Carry flag	1	1
ADDC	A,#data	Add immediate data to A with Carry flag	2	1
SUBB	A,Rn	Subtract register from A with Borrow	1	1
SUBB	A,direct	Subtract direct byte from A with Borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A with Borrow	1	1
SUBB	A,#data	Subtract immed data from A with Borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
INC	DPTR	Increment Data Pointer	1	2
DEC	A	Decrement Accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
MUL	AB	Multiply A & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1

Table 2. MCS®-51 Instruction Set Description (Continued)

Mnemonic		Description	Byte	Cyc
LOGICAL OPERATIONS				
ANL	A,Rn	AND register to Accumulator	1	1
ANL	A,direct	AND direct byte to Accumulator	2	1
ANL	A,@RI	AND indirect RAM to Accumulator	1	1
ANL	A,#data	AND immediate data to Accumulator	2	1
ANL	direct,A	AND Accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to Accumulator	1	1
ORL	A,direct	OR direct byte to Accumulator	2	1
ORL	A,@Ri	OR indirect RAM to Accumulator	1	1
ORL	A,#data	OR immediate data to Accumulator	2	1
ORL	direct,A	OR Accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive-OR register to Accumulator	1	1
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	1
XRL	A,@RI	Exclusive-OR indirect RAM to A	1	1
XRL	A,#data	Exclusive-OR immediate data to A	2	1
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	1
XRL	direct,#data	Exclusive-OR immediate data to direct	3	2
CLR	A	Clear Accumulator	1	1
CPL	A	Complement Accumulator	1	1

Mnemonic		Description	Byte	Cyc
LOGICAL OPERATIONS (Continued)				
RL	A	Rotate Accumulator Left	1	1
RLC	A	Rotate A Left through the Carry flag	1	1
RR	A	Rotate Accumulator Right	1	1
RRC	A	Rotate A Right through Carry flag	1	1
SWAP	A	Swap nibbles within the Accumulator	1	1
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	1
MOV	A,direct	Move direct byte to Accumulator	2	1
MOV	A,@RI	Move indirect RAM to Accumulator	1	1
MOV	A,#data	Move immediate data to Accumulator	2	1
MOV	Rn,A	Move Accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move Accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct,#data	Move immediate data to direct byte	3	2
MOV	@Ri,A	Move Accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2

Table 2. MCS®-51 Instruction Set Description (Continued)

Mnemonic		Description	Byte	Cyc
DATA TRANSFER (Continued)				
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	2
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to A	1	2
MOVC	A,@A+PC	Move Code byte relative to PC to A	1	2
MOVX	A,@Ri	Move External RAM (8-bit addr) to A	1	2
MOVX	A,@DPTR	Move External RAM (16-bit addr) to A	1	2
MOVX	@Ri,A	Move A to External RAM (8-bit addr)	1	2
MOVX	@DPTR,A	Move A to External RAM (16-bit) addr	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with Accumulator	1	1
XCH	A,direct	Exchange direct byte with Accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with A	1	1
XCHD	A,@Ri	Exchange low-order Digit ind RAM w A	1	1
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set Carry Flag	1	1
SETB	bit	Set direct Bit	2	1
CPL	C	Complement Carry Flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to Carry flag	2	2

Mnemonic		Description	Byte	Cyc
BOOLEAN VARIABLE MANIPULATION (Continued)				
ANL	C,/bit	AND complement of direct bit to Carry	2	2
ORL	C/bit	OR direct bit to Carry flag	2	2
ORL	C,/bit	OR complement of direct bit to Carry	2	2
MOV	C,/bit	Move direct bit to Carry flag	2	1
MOV	bit,C	Move Carry flag to direct bit	2	2
PROGRAM AND MACHINE CONTROL				
ACALL	addr11	Absolute Subroutine Call	2	2
LCALL	addr16	Long Subroutine Call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute Jump	2	2
LJMP	addr16	Long Jump	3	2
SJMP	rel	Short Jump (relative addr)	2	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if Accumulator is Zero	2	2
JNZ	rel	Jump if Accumulator is Not Zero	2	2
JC	rel	Jump if Carry flag is set	2	2
JNC	rel	Jump if No Carry flag	2	2
JB	bit,rel	Jump if direct Bit set	3	2
JNB	bit,rel	Jump if direct Bit Not set	3	2
JBC	bit,rel	Jump if direct Bit is set & Clear bit	3	2
CJNE	A,direct,rel	Compare direct to A & Jump if Not Equal	3	2
CJNE	A,#data,rel	Comp, immed, to A & Jump if Not Equal	3	2

Table 2. MCS®-51 Instruction Set Description (Continued)

Mnemonic	Description	Byte	Cyc
PROGRAM AND MACHINE CONTROL (Continued)			
CJNE Rn, #data,rel	Comp, immed, to reg & Jump if Not Equal	3	2
CJNE @Ri, #data, rel	Comp, immed, to ind. & Jump if Not Equal	3	2
DJNZ Rn,rel	Decrement register & Jump if Not Zero	2	2
DJNZ direct,rel	Decrement direct & Jump if Not Zero	3	2
NOP	No operation	1	1

Notes on data addressing modes:

Rn — Working register R0-R7

direct — 128 internal RAM locations, any I/O port, control or status register

@Ri — Indirect internal RAM location addressed by register R0 or R1

Notes on data addressing modes: (Continued)

#data — 8-bit constant included in instruction

#data16 — 16-bit constant included as bytes 2 & 3 of instruction

bit — 128 software flags, any I/O pin, controll or status bit

Notes on program addressing modes:

addr16 — Destination address for LCALL & LJMP may be anywhere within the 64-K program memory address space

Addr11 — Destination address for ACALL & AJMP will be within the same 2-K page of program memory as the first byte of the following instruction

rel — SJMP and all conditional jumps include an 8-bit offset byte, Range is +127 −128 bytes relative to first byte of the following instruction

All mnemonic copyrighted© Intel Corporation 1979

Timer/Counters

The 8044 contains two 16-bit counters which can be used for measuring time intervals, measuring pulse widths, counting events, generating precise periodic interrupt requests, and clocking the serial communications. Internally the Timers are clocked at 1/12 of the crystal frequency, which is the instruction cycle time. Externally the counters can run up to 500 KHz.

Interrupt System

External events and the real-time driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, a sophisticated multiple-source, two priority level, nested interrupt system is provided. Interrupt response latency ranges from 3 μsec to 7 μsec when using a 12 MHz clock.

All five interrupt sources can be mapped into one of the two priority levels. Each interrupt source can be enabled or disabled individually or the entire interrupt system can be enabled or disabled. The five interrupt sources are: Serial Interface Unit, Timer 1, Timer 2, and two external interrupts. The external interrupts can be either level or edge triggered.

Serial Interface Unit (SIU)

The Serial Interface Unit is used for HDLC/SDLC communications. It handles Zero Bit Insertion/Deletion, Flags automatic access recognization, and a 16-bit cyclic redundancy check. In addition it implements in hardware a subset of the SDLC protocol certain applications it is advantageous to have the CPU control the reception or transmission of every single frame. For this reason the SIU has two modes of operation: "AUTO" and "FLEXIBLE" (or "NON-AUTO"). It is in the AUTO mode that the SIU responds to SDLC frames without CPU intervention; whereas, in the FLEXIBLE mode the reception or transmission of every single frame will be under CPU control.

There are three control registers and eight parameter registers that are used to operate the serial interface. These registers are shown in Figure 5 and Figure 6. The control register set the modes of operation and provide status information. The eight parameter registers buffer the station address, receive and transmit control bytes, and point to the on-chip transmit and receive buffers.

Data to be received or transmitted by the SIU must be buffered anywhere within the 192 bytes of on-chip RAM. Transmit and receive buffers are not allowed to "wrap around" in RAM; a "buffer end" is generated after address 191 is reached.

REGISTER NAMES	SYMBOLIC ADDRESS	BIT ADDRESS	BYTE ADDRESS
B REGISTER	B	247 through 240	240 (F0H)
ACCUMULATOR	ACC	231 through 224	224 (E0H)
*THREE BYTE FIFO	FIFO		223 (DFH)
	FIFO		222 (DEH)
	FIFO		221 (DDH)
TRANSMIT BUFFER START	TBS		220 (DCH)
TRANSMIT BUFFER LENGTH	TBL		219 (DBH)
TRANSMIT CONTROL BYTE	TCB		218 (DAH)
* SIU STATE COUNTER	SIUST		217 (D9H)
SEND COUNT RECEIVE COUNT	NSNR	223 through 216	216 (D8H)
PROGRAM STATUS WORD	PSW	215 through 208	208 (D0H)
*DMA COUNT	DMA CNT		207 (CFH)
STATION ADDRESS	STAD		206 (CEH)
RECEIVE FIELD LENGTH	RFL		205 (CDH)
RECEIVE BUFFER START	RBS		204 (CCH)
RECEIVE BUFFER LENGTH	RBL		203 (CBH)
RECEIVE CONTROL BYTE	RCB		202 (CAH)
SERIAL MODE	SMD		201 (C9H)
STATUS REGISTER	STS	207 through 200	200 (C8H)
INTERRUPT PRIORITY CONTROL	IP	191 through 184	184 (B8H)
PORT 3	P3	183 through 176	176 (B0H)
INTERRUPT ENABLE CONTROL	IE	175 through 168	168 (A8H)
PORT 2	P2	167 through 160	160 (A0H)
PORT 1	P1	151 through 144	144 (90H)
TIMER HIGH 1	TH1		141 (8DH)
TIMER HIGH 0	TH0		140 (8CH)
TIMER LOW 1	TL1		139 (8BH)
TIMER LOW 0	TL0		138 (8AH)
TIMER MODE	TMOD		137 (89H)
TIMER CONTROL	TCON	143 through 136	136 (88H)
DATA POINTER HIGH	DPH		131 (83H)
DATA POINTER LOW	DPL		130 (82H)
STACK POINTER	SP		129 (81H)
PORT 0	P0	135 through 128	128 (80H)

SFR's CONTAINING DIRECT ADDRESSABLE BITS

231663-6

NOTE:
*ICE Support Hardware registers. Under normal operating conditions there is no need for the CPU to access these registers.

Figure 5. Mapping of Special Function Registers

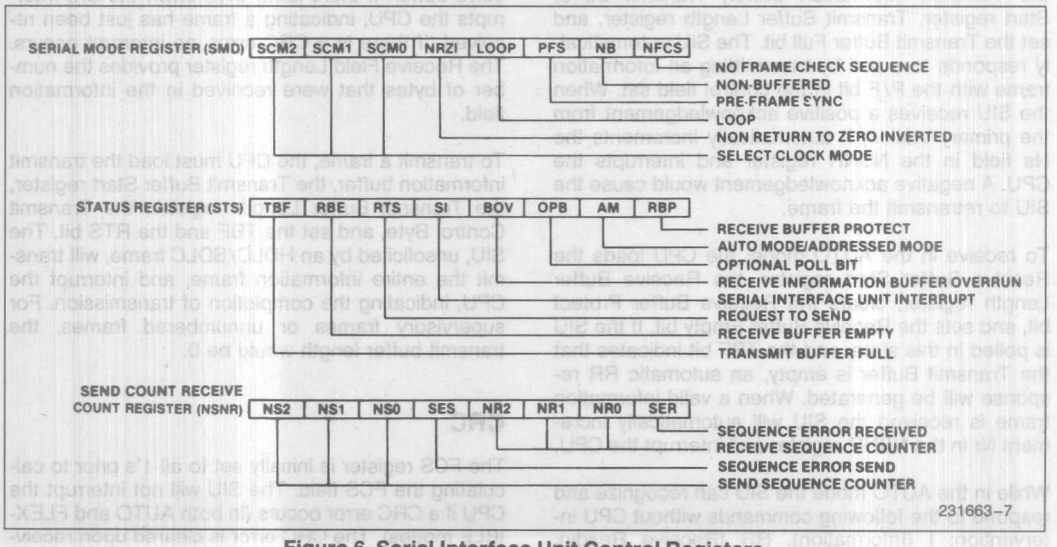


Figure 6. Serial Interface Unit Control Registers

With the addition of only a few bytes of code, the 8044's frame size is not limited to the size of its internal RAM (192 bytes), but rather by the size of external buffer with no degradation of the RUPI's features (e.g. NRZI, zero bit insertion/deletion, address recognition, cyclic redundancy check). There is a special function register called SIUST whose contents dictates the operation of the SIU. At low data rates, one section of the SIU (the Byte Processor) performs no function during known intervals. For a given data rate, these intervals (stand-by mode) are fixed. The above characteristics make it possible to program the CPU to move data to/from external RAM and to force the SIU to perform some desired hardware tasks while transmission or reception is taking place. With these modifications, external RAM can be utilized as a transmit and received buffer instead of the internal RAM.

AUTO Mode

In the AUTO mode the SIU implements in hardware a subset of the SDLC protocol such that it responds to many SDLC frames without CPU intervention. All AUTO mode responses to the primary station will comform to IBM's SDLC definition. The advantages of the AUTO mode are that less software is required to implement a secondary station, and the hardware generated response to polls is much faster than doing it in software. However, the Auto mode can not be used at a primary station.

To transmit in the AUTO mode the CPU must load the Transmit Information Buffer, Transmit Buffer Start register, Transmit Buffer Length register, and set the Transmit Buffer Full bit. The SIU automatically responds to a poll by transmitting an information frame with the P/F bit in the control field set. When the SIU receives a positive acknowledgement from the primary station, it automatically increments the Ns field in the NSNR register and interrupts the CPU. A negative acknowledgement would cause the SIU to retransmit the frame.

To receive in the AUTO mode, the CPU loads the Receive Buffer Start register, the Receive Buffer Length register, clears the Receive Buffer Protect bit, and sets the Receive Buffer Empty bit. If the SIU is polled in this state, and the TBF bit indicates that the Transmit Buffer is empty, an automatic RR response will be generated. When a valid information frame is received the SIU will automatically increment Nr in the NSNR register and interrupt the CPU.

While in the AUTO mode the SIU can recognize and respond to the following commands without CPU intervention: I (Information), RR (Receive Ready), RNR (Receive Not Ready), REJ (Reject), and UP (Unnumbered Poll). The SIU can generate the following responses without CPU intervention: I (Information), RR (Receive Ready), and RNR (Receive Not Ready).

When the Receive Buffer Empty bit (RBE) indicates that the Receive Buffer is empty, the receiver is enabled, and when the RBE bit indicates that the Receive Buffer is full, the receiver is disabled. Assuming that the Receiver Buffer is empty, the SIU will respond to a poll with an I frame if the Transmit Buffer is full. If the Transmit Buffer is empty, the SIU will respond to a poll with a RR command if the Receive Buffer Protect bit (RBP) is cleared, or an RNR command if RBP is set.

FLEXIBLE (or NON-AUTO) Mode

In the FLEXIBLE mode all communications are under control of the CPU. It is the CPU's task to encode and decode control fields, manage acknowledgements, and adhere to the requirements of the HDLC/SDLC protocols. The 8044 can be used as a primary or a secondary station in this mode.

To receive a frame in the FLEXIBLE mode, the CPU must load the Receive Buffer Start register, the Receive Buffer Length register, clear the Receive Buffer Protect bit, and set the Receive Buffer Empty bit. If a valid opening flag is received and the address field matches the byte in the Station Address register or the address field contains a broadcast address, the 8044 loads the control field in the receive control byte register, and loads the I field in the receive buffer. If there is no CRC error, the SIU interrupts the CPU, indicating a frame has just been received. If there is a CRC error, no interrupt occurs. The Receive Field Length register provides the number of bytes that were received in the information field.

To transmit a frame, the CPU must load the transmit information buffer, the Transmit Buffer Start register, the Transmit Buffer Length register, the Transmit Control Byte, and set the TBF and the RTS bit. The SIU, unsolicited by an HDLC/SDLC frame, will transmit the entire information frame, and interrupt the CPU, indicating the completion of transmission. For supervisory frames or unnumbered frames, the transmit buffer length would be 0.

CRC

The FCS register is initially set to all 1's prior to calculating the FCS field. The SIU will not interrupt the CPU if a CRC error occurs (in both AUTO and FLEXIBLE modes). The CRC error is cleared upon receiving of an opening flag.

Frame Format Options

In addition to the standard SDLC frame format, the 8044 will support the frames displayed in Figure 7. The standard SDLC frame is shown at the top of this figure. For the remaining frames the information field will incorporate the control or address bytes and the frame check sequences; therefore these fields will be stored in the Transmit and Receive buffers. For example, in the non-buffered mode the third byte is treated as the beginning of the information field. In the non-addressed mode, the information field begins after the opening flag. The mode bits to set the frame format options are found in the Serial Mode register and the Status register.

FRAME OPTION	NFCS	NB	AM[1]	FRAME FORMAT
Standard SDLC NON-AUTO Mode	0	0	0	F A C I FCS F
Standard SDLC AUTO Mode	0	0	1	F A C I FCS F
Non-Buffered Mode NON-AUTO Mode	0	1	1	F A I FCS F
Non-Addressed Mode NON-AUTO Mode	0	1	0	F I FCS F
No FCS Field NON-AUTO Mode	1	0	0	F A C I F
No FCS Field AUTO Mode	1	0	1	F A C I F
No FCS Field Non-Buffered Mode NON-AUTO Mode	1	1	1	F A I F
No FCS Field Non-Addressed Mode NON-AUTO Mode	1	1	0	F I F

Mode Bits:
AM — "AUTO" Mode/Addressed Mode
NB — Non-Buffered Mode
NFCS — No FCS Field Mode

Key to Abbreviations:
F = Flag (01111110)
A = Address Field
C = Control Field
I = Information Field
FCS= Frame Check Sequence

Note 1:
The AM bit function is controlled by the NB bit. When NB = 0, AM becomes AUTO mode select, when NB = 1, AM becomes Address mode select.

Figure 7. Frame Format Options

To realize an extended control field or an extended address field using the HDLC protocol, the FLEXIBLE mode must be used. For an extended control field, the SIU is programmed to be in the non-buffered mode. The extended control field will be the first and second bytes in the Receive and Transmit Buffers. For extended addressing the SIU is placed in the non-addressed mode. In this mode the CPU must implement the address recognition for received frames. The addressing field will be the initial bytes in the Transmit and Receive buffers followed by the control field.

The SIU can transmit and receive only frames which are multiples of 8 bits. For frames received with other than 8-bit multiples, a CRC error will cause the SIU to reject the frame.

SDLC Loop Networks

The SIU can be used in an SDLC loop as a secondary or primary station. When the SIU is placed in the Loop mode it receives the data on pin 10 and transmits the data one bit time delayed on pin 11. It can also recognize the Go ahead signal and change it into a flag when it is ready to transmit. As a secondary station the SIU can be used in the AUTO or FLEXIBLE modes. As a primary station the FLEXIBLE mode is used; however, additional hardware is required for generating the Go Ahead bit pattern. In the Loop mode the maximum data rate is 1 Mbps clocked or 375 Kpbs self-clocked.

SDLC Multidrop Networks

The SIU can be used in a SDLC non-loop configuration as a secondary or primary station. When the SIU is placed in the non-loop mode, data is received and transmitted on pin 11, and pin 10 drives a tri-state buffer. In non-loop mode, modem interface pins, RTS and CTS, become available.

Data Clocking Options

The 8044's serial port can operate in an externally clocked or self clocked system. A clocked system provides to the 8044 a clock synchronization to the data. A self-clocked system uses the 8044's on-chip Digital Phase Locked Loop (DPLL) to recover the clock from the data, and clock this data into the Serial Receive Shift Register.

In this mode, a clock synchronized with the data is externally fed into the 8044. This clock may be generated from an External Phase Locked Loop, or possibly supplied along with the data. The 8044 can 2.4 Mbps.

This self clocked mode allows data transfer without a common system data clock. An on-chip Digital Phase Locked Loop is employed to recover the data clock which is encoded in the data stream. The DPLL will converge to the nominal bit center within eight bit transitions, worst case. The DPLL requires a reference clock of either 16 times (16x) or 32 times (32x) the data rate. This reference clock may be externally applied or internally generated. When internally generated either the 8044's internal logic clock (crystal frequency divided by two) or the timer 1 overflow is used as the reference clock. Using the internal timer 1 clock the data rates can vary from 244 to 62.5 Kbps. Using the internal logic clock at a 16x sampling rate, receive data can either be 187.5 Kbps, or 375 Kbps. When the reference clock for the DPLL is externally applied the data rates can vary from 0 to 375 Kbps at a 16x sampling rate.

To aid in a Phase Locked Loop capture, the SIU has a NRZI (Non Return to Zero Inverted) data encoding and decoding option. Additionally the SIU has a preframe sync option that transmits two bytes of alternating 1's and 0's to ensure that the receive station DPLL will be synchronized with the data by the time it receives the opening flag.

Control and Status Registers

There are three SIU Control and Status Registers:

Serial Mode Register (SMD)

Status/Command Register (STS)

Send/Receive Count Register (NSNR)

The SMD, STS, and NSNR, registers are all cleared by system reset. This assures that the SIU will power up in an idle state (neither receiving nor transmitting).

These registers and their bit assignments are described below.

SMD: Serial Mode Register (byte-addressable)

Bit 7:	6	5	4	3	2	1	0
SCM2	SCM1	SCM0	NRZI	LOOP	PFS	NB	NFCS

The Serial Mode Register (Address C9H) selects the operational modes of the SIU. The 8044 CPU can both read and write SMD. The SIU can read SMD but cannot write to it. To prevent conflict between CPU and SIU access to SMD, the CPU should write SMD only when the Request To Send (RTS) and

Receive Buffer Empty (RBE) bits (in the STS register) are both false (0). Normally, SMD is accessed only during initialization.

The individual bits of the Serial Mode Register are as follows:

Bit #	Name	Description
SMD.0	NFCS	No FCS field in the SDLC frame.
SMD.1	NB	Non-Buffered mode. No control field in the SDLC frame.
SMD.2	PFS	Pre-Frame Sync mode. In this mode, the 8044 transmits two bytes before the first flag of a frame, for DPLL synchronization. If NRZI is enabled, 00H is sent; otherwise, 55H is sent. In either case, 16 preframe transitions are guaranteed.
SMD.3	LOOP	Loop configuration.
SMD.4	NRZI	NRZI coding option. If bit = 1, NRZI coding is used. If bit = 0, then it is straight binary (NRZ).
SMD.5	SCM0	Select Clock Mode—Bit 0
SMD.6	SCM1	Select Clock Mode—Bit 1
SMD.7	SCM2	Select Clock Mode—Bit 2

The SCM bits decode as follows:

SCM				Data Rate
2	1	0	Clock Mode	(Bits/sec)*
0	0	0	Externally clocked	0-2.4M**
0	0	1	Reserved	
0	1	0	Self clocked, timer overflow	244-62.5K
0	1	1	Reserved	
1	0	0	Self clocked, external 16x	0-375K
1	0	1	Self clocked, external 32x	0-187.5K
1	1	0	Self clocked, internal fixed	375K
1	1	1	Self clocked, internal fixed	187.5K

NOTES:
*Based on a 12 Mhz crystal frequency
**0-1 M bps in loop configuration

STS: Status/Command Register (bit-addressable)

Bit: 7 6 5 4 3 2 1 0

TBF	RBE	RTS	SI	BOV	OPB	AM	RBP

The Status/Command Register (Address C8H) provides operational control of the SIU by the 8044 CPU, and enables the SIU to post status information for the CPU's access. The SIU can read STS, and can alter certain bits, as indicated below. The CPU can both read and write STS asynchronously. However, 2-cycle instructions that access STS during both cycles ('JBC/B, REL' and 'MOV/B, C.') should not be used, since the SIU may write to STS between the two CPU accesses.

The individual bits of the Status/Command Register are as follows:

Bit #	Name	Description
STS.0	RBP	Receive Buffer Protect. Inhibits writing of data into the receive buffer. In AUTO mode, RBP forces an RNR response instead of an RR.
STS.1	AM	AUTO Mode/Addressed Mode. Selects AUTO mode where AUTO mode is allowed. If NB is true, (= 1), the AM bit selects the addressed mode. AM may be cleared by the SIU.
STS.2	OPB	Optional Poll Bit. Determines whether the SIU will generate an AUTO response to an optional poll (UP with P = 0). OPM may be set or cleared by the SIU.
STS.3	BOV	Receive Buffer Overrun. BOV may be set or cleared by the SIU.
STS.4	SI	SIU Interrupt. This is one of the five interrupt sources to the CPU. The vector location = 23H. SI may be set by the SIU. It should be cleared by the CPU before returning from an interrupt routine.
STS.5	RTS	Request To Send. Indicates that the 8044 is ready to transmit or is transmitting. RTS may be read or written by the CPU. RTS may be read by the SIU, and in AUTO mode may be written by the SIU.
STS.6	RBE	Receive Buffer Empty. RBE can be thought of as Receive Enable. RBE is set to one by the CPU when it is ready to receive a frame, or has just read the buffer, and to zero by the SIU when a frame has been received.
STS.7	TBF	Transmit Buffer Full. Written by the CPU to indicate that it has filled the transmit buffer. TBF may be cleared by the SIU.

NSNR: Send/Receive Count Register (bit-addressable)

Bit:	7	6	5	4	3	2	1	0
	NS2	NS1	NS0	SES	NR2	NR1	NR0	SER

The Send/Receive Count Register (Address D8H) contains the transmit and receive sequence numbers, plus tally error indications. The SIU can both read and write NSNR. The 8044 CPU can both read and write NSNR asynchronously. However, 2-cycle instructions that access NSNR during both cycles ('JBC /B, REL,' and 'MOV /B,C') should not be used, since the SIU may write to NSMR between the two 8044 CPU accesses.

The individual bits of the Send/Receive Count Register are as follows:

Bit #	Name	Description
NSNR.0	SER	Receive Sequence Error: NS (P) ≠ NR (S)
NSNR.1	NR0	Receive Sequence Counter—Bit 0
NSNR.2	NR1	Receive Sequence Counter—Bit 1
NSNR.3	NR2	Receive Sequence Counter—Bit 2
NSNR.4	SES	Send Sequence Error: NR (P) ≠ NS (S) and NR (P) ≠ NS (S) + 1
NSNR.5	NS0	Send Sequence Counter—Bit 0
NSNR.6	NS1	Send Sequence Counter—Bit 1
NSNR.7	NS2	Send Sequence Counter—Bit 2

Parameter Registers

There are eight parameter registers that are used in connection with SIU operation. All eight registers may be read or written by the 8044 CPU. RFL and RCB are normally loaded by the SIU.

The eight parameter registers are as follows:

STAD: Station Address Register (byte-addressable)

The Station Address register (Address CEH) contains the station address. To prevent acess conflict, the CPU should access STAD only when the SIU is idle (RTS = 0 and RBE = 0). Normally, STAD is accessed only during initialization.

TBS: Transmit Buffer Start Address Register (byte-addressable)

The Transmit Buffer Start address register (Address DCH) points to the location in on-chip RAM for the beginning of the I-field of the frame to be transmitted. The CPU should access TBS only when the SIU is not transmitting a frame (when TBF = 0).

TBL: Transmit Buffer Length Register (byte = addressable)

The Transmit Buffer Length register (Address DBH) contains the length (in bytes) of the I-field to be transmitted. A blank I-field (TBL = 0) is valid. The CPU should access TBL only when the SIU is not transmitting a frame (when TBF = 0).

NOTE:
The transmit and receive buffers are not allowed to "wrap around" in the on-chip RAM. A "buffer end" is automatically generated if address 191 (BFH) is reached.

TCB: Transmit Control Byte Register (byte-addressable)

The Transmit Control Byte register (Address DAH) contains the byte which is to be placed in the control field of the transmitted frame, during NON-AUTO mode transmission. The CPU should access TCB only when the SIU is not transmitting a frame (when TBF = 0). The N_S and N_R counters are not used in the NON-AUTO mode.

RBS: Receive Buffer Start Address Register (byte-addressable)

The Receive Buffer Start address register (Address CCH) points to the location in on-chip RAM where the beginning of the I-field of the frame being received is to be stored. The CPU should write RBS only when the SIU is not receiving a frame (when RBE = 0).

RBL: Receive Buffer Length Register (byte-addressable)

The Receive Buffer Length register (Address CBH) contains the length (in bytes) of the area in on-chip RAM allocated for the received I-field. RBL = 0 is valid. The CPU should write RBL only when RBE = 0.

RFL: Receive Field Length Register (byte-addressable)

The Receive Field Length register (Address CDH) contains the length (in bytes) of the received I-field that has just been loaded into on-chip RAM. RFL is loaded by the SIU. RFL = 0 is valid. RFL should be accessed by the CPU only when RBE = 0.

RCB: Receive Control Byte Register (byte-addressable)

The Received Control Byte register (Address CAH) contains the control field of the frame that has just been received. RCB is loaded by the SIU. The CPU can only read RCB, and should only access RCB when RBE = 0.

ICE Support

The 8044 In-Circuit Emulator (ICE-44) allows the user to exercise the 8044 application system and monitor the execution of instructions in real time.

The emulator operates with Intel's Intellec™ development system. The development system interfaces with the user's 8044 system through an in-cable buffer box. The cable terminates in a 8044 pin-compatible plug, which fits into the 8044 socket in the user's system. With the emulator plug in place, the user can excercise his system in real time while collecting up to 255 instruction cycles of real-time data. In addition, he can single-step the program.

Static RAM is available (in the in-cable buffer box) to emulate the 8044 internal and external program memory and external data memory. The designer can display and alter the contents of the replacement memory in the buffer box, the internal data memory, and the internal 8044 registers, including the SFR's.

SIUST: SIU State Counter (byte-addressable)

The SIU State Counter (Address D9H) reflects the state of the internal logic which is under SIU control. Therefore, care must be taken not to write into this register. This register provides a useful means for debugging 8044 receiver problem.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature.....−65°C to −150°C
Voltage on $\overline{EA}$, VPP Pin to VSS...−0.5V to −21.5V
Voltage on Any Other Pin to VSS ....−0.5V to −7V
Power Dissipation 2W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, VCC = 5V ± 10%, VSS = 0V

Symbol	Parameter		Min	Max	Unit	Test Conditions
VIL	Input Low Voltage (Except $\overline{EA}$ Pin of 8744H)		−0.5	0.8	V	
VIL1	Input Low Voltage to $\overline{EA}$ Pin of 8744H		0	0.8	V	
VIH	Input High Voltage (Except XTAL2, RST)		2.0	VCC + 0.5	V	
VIH1	Input High Voltage to XTAL2, RST		2.5	VCC + 0.5	V	XTAL1 = VSS
VOL	Output Low Voltage (Ports 1, 2, 3)*			0.45	V	IOL = 1.6mA
VOL1	Output Low Voltage (Port 0,ALE,$\overline{PSEN}$)*					
		8744H		0.60 0.45	V V	IOL = 3.2 mA IOL = 2.4 mA
		8044AH/8344AH		0.45	V	IOL = 3.2 mA
VOH	Output High Voltage (Ports 1, 2, 3)		2.4		V	IOH = −80 μA
VOH1	Output High Voltage (Port 0 in External Bus Mode, ALE, $\overline{PSEN}$)		2.4		V	IOH = −400 μA
IIL	Logical 0 Input Current (Ports 1, 2, 3)			−500	μA	Vin = 0.45V
IIL1	Logical 0 Input Current to $\overline{EA}$ Pin of 8744H only			−15	mA	
IIL2	Logical 0 Input Current (XTAL2)			−3.6	mA	Vin = 0.45V
ILI	Input Leakage Current (Port 0) 8744H 8044AH/8344AH			 ±100 ±10	 μA μA	 0.45 < Vin < VCC 0.45 < Vin < VCC
IIH	Logical 1 Input Current to $\overline{EA}$ Pin of 8744H			500	μA	
IIH1	Input Current to RST to Activate Reset			500	μA	Vin < (VCC − 1.5V)
ICC	Power Supply Current: 8744H 8044AH/8344AH			 285 170	 mA mA	All Outputs Disconnected: $\overline{EA}$ = VCC
CIO	Pin Capacitance			10	pF	Test Freq. = 1MHz(1)

***NOTES:**
1. Sampled not 100% tested. T_A = 25°C.
2. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the VOLs of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pin when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

A.C. CHARACTERISTICS

T_A = 0°C to +70°C, VCC = 5V ± 10%, VSS = 0V, Load Capacitance for Port 0, ALE, and PSEN = 100 pF, Load Capacitance for All Other Outputs = 80 pF

EXTERNAL PROGRAM MEMORY CHARACTERISTICS

Symbol	Parameter	12 MHz Osc		Variable Clock 1/TCLCL = 3.5 MHz to 12 MHz		Unit
		Min	Max	Min	Max	
TLHLL	ALE Pulse Width	127		2TCLCL-40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL-40		ns
TLLAX[1]	Address Hold After ALE Low	48		TCLCL-35		ns
TLLIV	ALE Low to Valid Instr in					ns
	8744H		183		4TCLCL-150	
	8044AH/8344AH		233		4TCLCL-100	
TLLPL	ALE Low to $\overline{\text{PSEN}}$ Low	58		TCLCL-25		ns
TPLPH	$\overline{\text{PSEN}}$ Pulse Width					
	8744H	190		3TCLCL-60		ns
	8044AH/8344AH	215		3TCLCL-35		ns
TPLIV	$\overline{\text{PSEN}}$ Low to Valid Instr in					
	8744H		100		3TCLCL-150	ns
	8044AH/8344AH		125		3TCLCL-125	ns
TPXIX	Input Instr Hold After $\overline{\text{PSEN}}$	0		0		ns
TPXIZ[2]	Input Instr Float After $\overline{\text{PSEN}}$		63		TCLCL-20	ns
TPXAV[2]	$\overline{\text{PSEN}}$ to Address Valid	75		TCLCL-8		ns
TAVIV	Address to Valid Instr in					
	8744H		267		5TCLCL-150	ns
	8044AH/8344AH		302		5TCLCL-115	ns
TAZPL	Address Float to $\overline{\text{PSEN}}$	−25		−25		ns

NOTES:
1. TLLAX for access to program memory is different from TLLAX for data memory.
2. Interfacing RUPI-44 devices with float times up to 75ns is permissible. This limited bus contention will not cause any damage to Port 0 drivers.

EXTERNAL DATA MEMORY CHARACTERISTICS

Symbol	Parameter	12 MHz Osc		Variable Clock 1/TCLCL = 3.5 MHz to 12 MHz		Unit
		Min	Max	Min	Max	
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL-100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL-100		ns
TLLAX	Address Hold after ALE	48		TCLCL-35		ns
TRLDV	$\overline{RD}$ Low to Valid Data in		252		5TCLCL-165	ns
TRHDX	Data Hold After $\overline{RD}$	0		0		ns
TRHDZ	Data Float After $\overline{RD}$		97		2TCLCL-70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL-150	ns
TAVDV	Address to Valid Data In		585		9TCLCL-165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL-50	3TLCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL-130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition					
	8744H	13		TCLCL-70		ns
	8044AH/8344AH	23		TCLCL-60		ns
TQVWH	Data Setup Before $\overline{WR}$ High	433		7TCLCL-150		ns
TWHQX	Data Held After $\overline{WR}$	33		TCLCL-50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		25		25	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High					
	8744H	33	133	TCLCL-50	TCLCL+50	ns
	8044AH/8344AH	43	123	TCLCL-40	TCLCL+50	ns

NOTE:
1. TLLAX for access to program memory is different from TLLAX for access data memory.

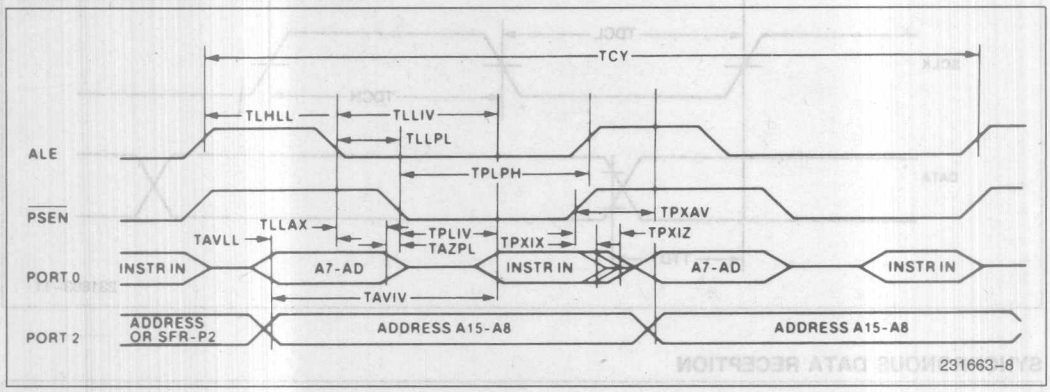
Serial Interface Characteristics

Symbol	Parameter	Min	Max	Unit
TDCY	Data Clock	420		ns
TDCL	Data Clock Low	180		ns
TDCH	Data Clock High	100		ns
tTD	Transmit Data Delay		140	ns
tDSS	Data Setup Time	40		ns
tDHS	Data Hold Time	40		ns

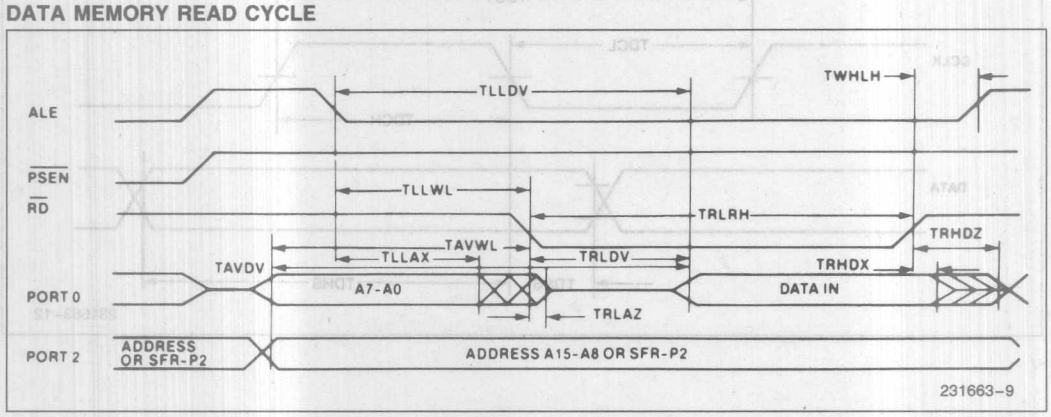
WAVEFORMS

Memory Access

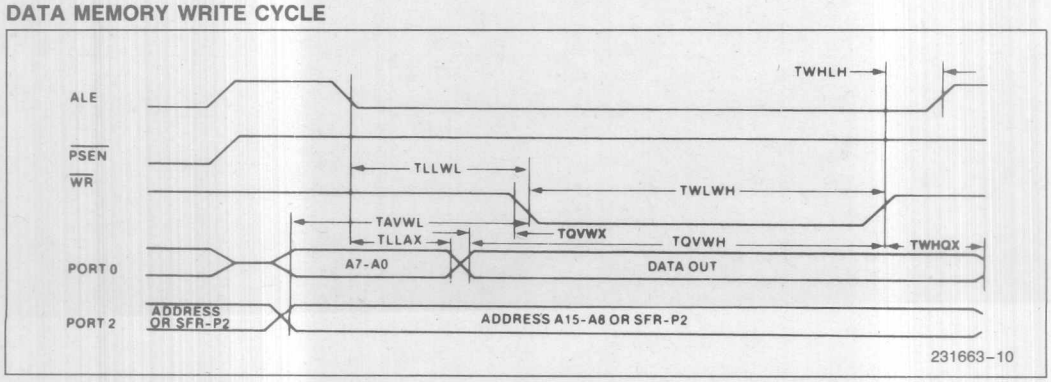
PROGRAM MEMORY READ CYCLE



DATA MEMORY READ CYCLE

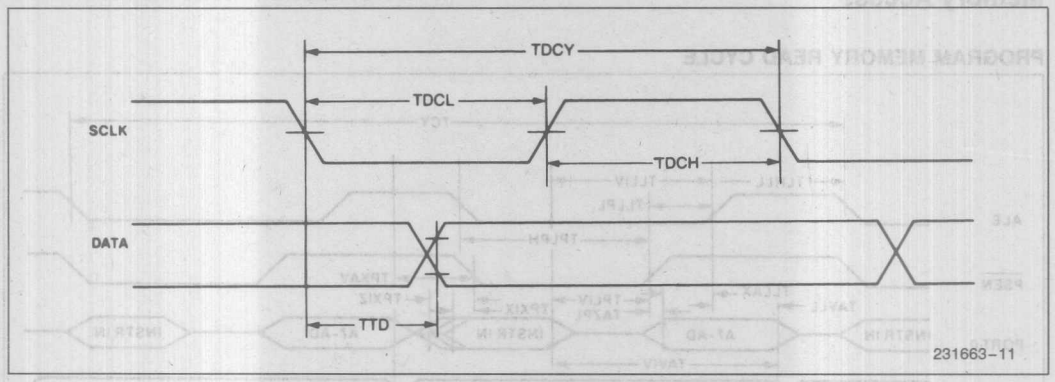


DATA MEMORY WRITE CYCLE

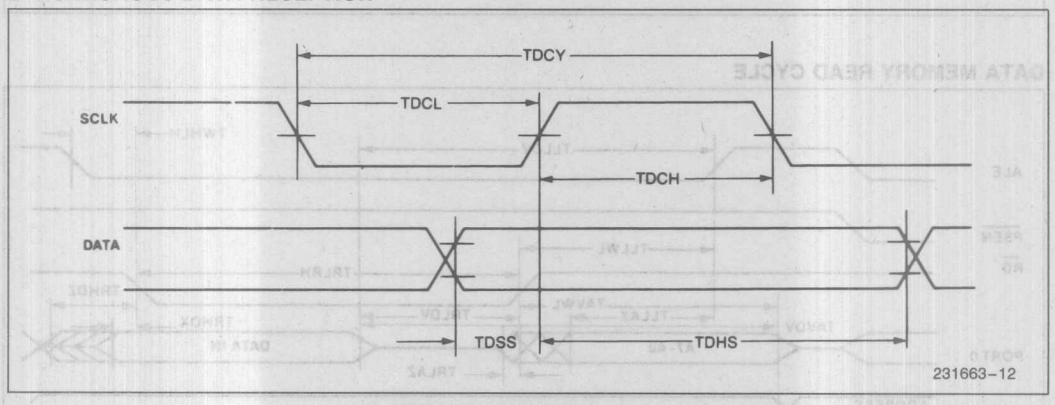


SERIAL I/O WAVEFORMS

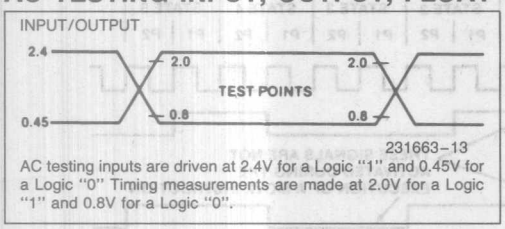
SYNCHRONOUS DATA TRANSMISSION



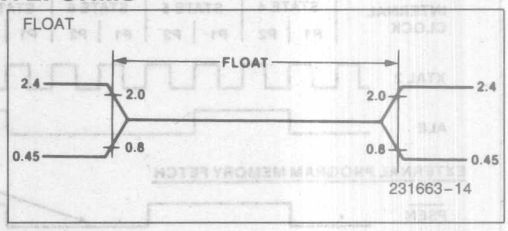
SYNCHRONOUS DATA RECEPTION



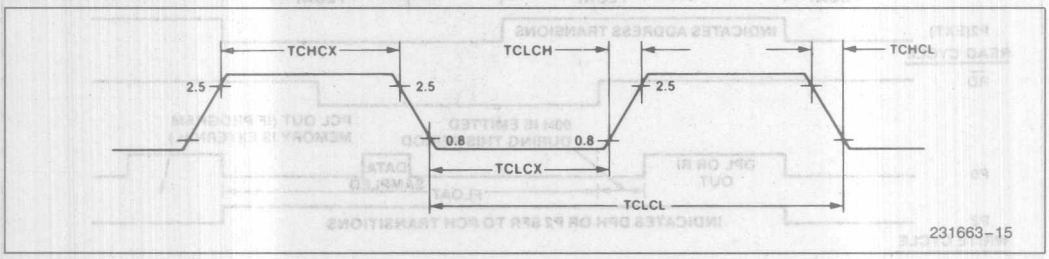
AC TESTING INPUT, OUTPUT, FLOAT WAVEFORMS



AC testing inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0" Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

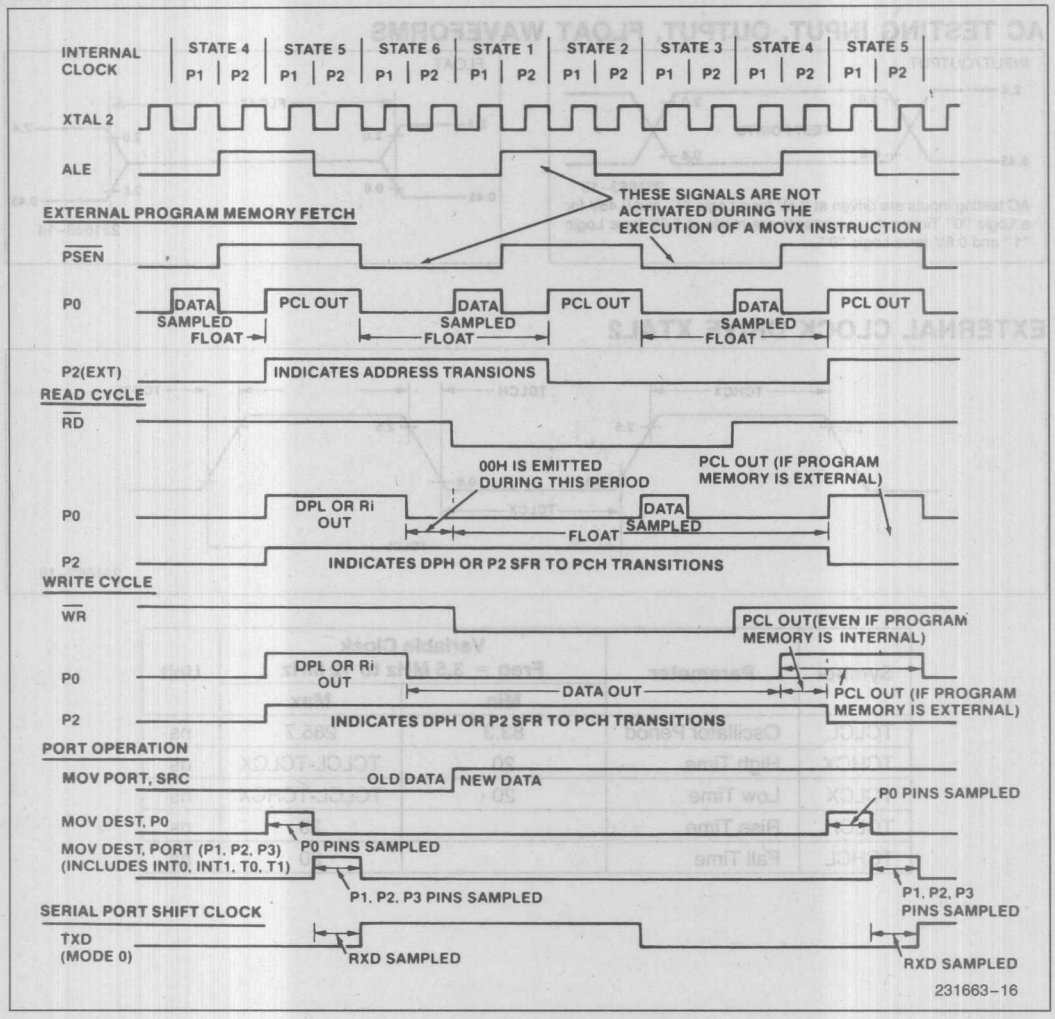


EXTERNAL CLOCK DRIVE XTAL2



Symbol	Parameter	Variable Clock Freq = 3.5 MHz to 12 MHz		Unit
		Min	Max	
TCLCL	Oscillator Period	83.3	285.7	ns
TCHCX	High Time	20	TCLCL-TCLCX	ns
TCLCX	Low Time	20	TCLCL-TCHCX	ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

CLOCK WAVEFORMS



This diagram indicates when signals are clocked internally. The time it takes the signals to propagate to the pins, however, ranges from 25 to 125 ns. This propagation delay is dependent on variables such as temperature and pin loading. Propagation also varies from output to output and component to component. Typically though, (T_A = 25°C, fully loaded) RD and WR propagation delays are approximately 50 ns. The other signals are typically 85 ns. Propagation delays are incorporated in the AC specifications.

8744H EPROM CHARACTERISTICS

Erasure Characteristics

Erasure of the 8744H Program Memory begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 Ångstroms. Since sunlight and fluorescent lighting have wavelengths in this range, constant exposure to these light sources over an extended period of time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause unintentional erasure. If an application subjects the 8744H to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Ångstroms) to an integrated dose of at least 15 W-sec/cm^2 rating for 20 to 30 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Programming the EPROM

To be programmed, the 8744H must be running with a 4 to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate registers.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0-P2.3 of Port 2, while the data byte is applied to Port 0. Pins P2.4-P2.6 and $\overline{PSEN}$ should be held low, and P2.7 and RST high. (These are all TTL levels except RST, which requires 2.5V for high.) $\overline{EA}$/VPP is held normally high, and is pulsed to +21V. While EA/VPP is at 21V, the ALE/$\overline{PROG}$ pin, which is normally being held high, is pulsed low for 50 msec. Then EA/VPP is returned to high. This is illustrated in Figure 8. Detailed timing specifications are provided in the EPROM Programming and Verification Characteristics section of this data sheet.

Program Memory Security

The program memory security feature is developed around a "security bit" in the 8744H EPROM array. Once this "hidden bit" is programmed, electrical access to the contents of the entire program memory array becomes impossible. Activation of this feature is accomplished by programming the 8744H as described in "Programming the EPROM" with the exception that P2.6 is held at a TTL high rather than a TTL low. In addition, Port 1 and P2.0-P2.3 may be in any state. Figure 9 illustrates the security bit programming configuration. Deactivating the security feature, which again allows programmability of the EPROM, is accomplished by exposing the EPROM to ultraviolet light. This exposure, as described in "Erasure Characteristics," erases the entire EPROM array. Therefore, attempted retrieval of "protected code" results in its destruction.

Program Verification

Program Memory may be read only when the "security feature" has not been activated. Refer to Figure 10 for Program Verification setup. To read the Program Memory, the following procedure can be used. The unit must be running with a 4 to 6 MHz oscillator. The address of a Program Memory location to be read is applied to Port 1 and pins P2.0-P2.3 of Port 2. Pins P2.4-P2.6 and $\overline{PSEN}$ are held at TTL low, while the ALE/$\overline{PROG}$, RST, and $\overline{EA}$/VPP pins are held at TTL high. (These are all TTL levels except RST, which requires 2.5V for high.) Port 0 will be the data output lines. P2.7 can be used as a read strobe. While P2.7 is held high, the Port 0 pins float. When P2.7 is strobed low, the contents of the addressed location will appear at Port 0. External pullups (e.g., 10K) are required on Port 0 during program verification.

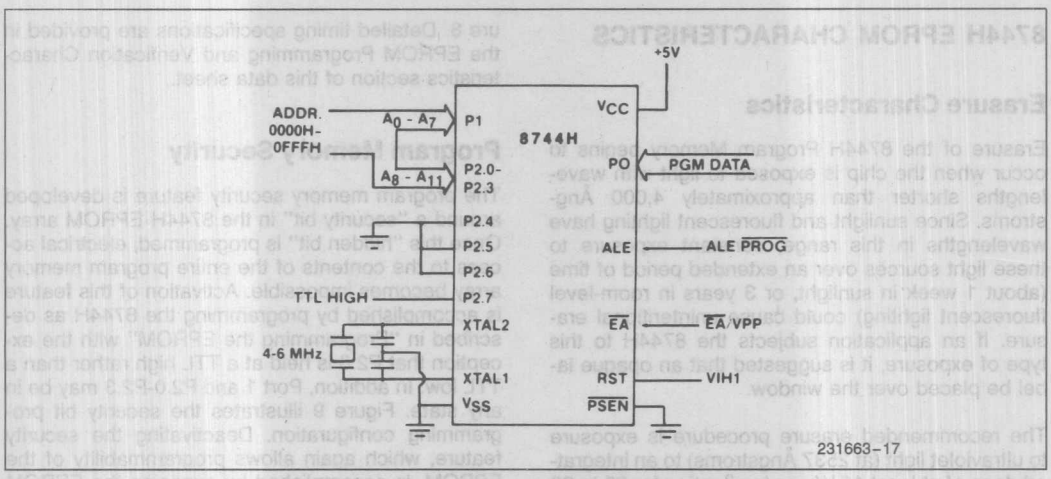


Figure 8. Programming Configuration

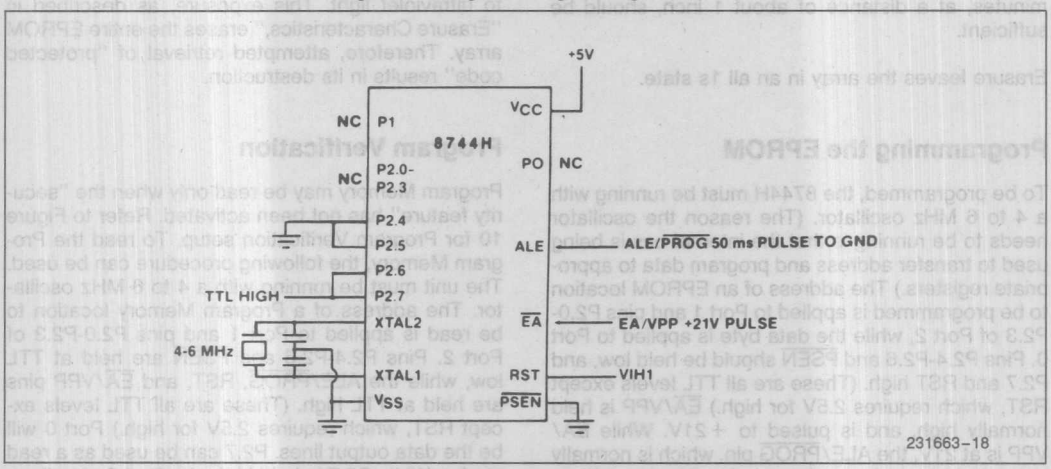


Figure 9. Security Bit Programming Configuration

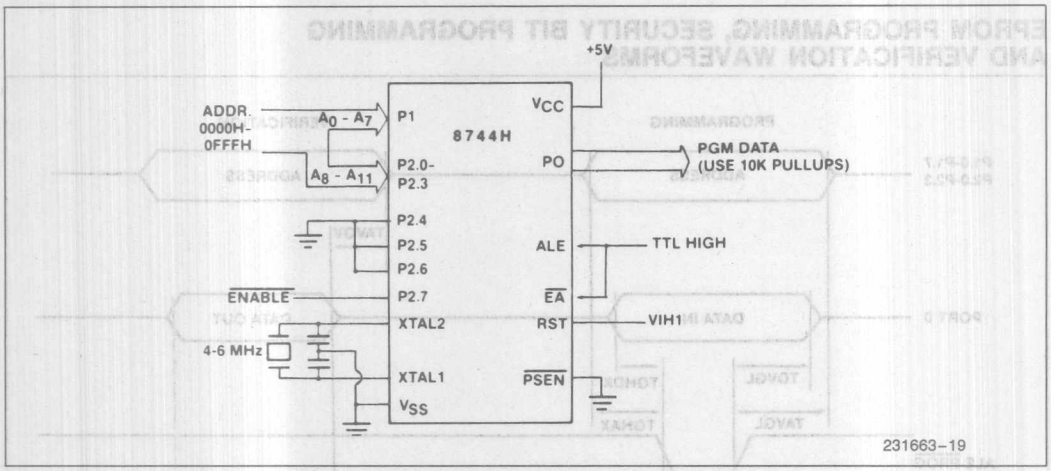


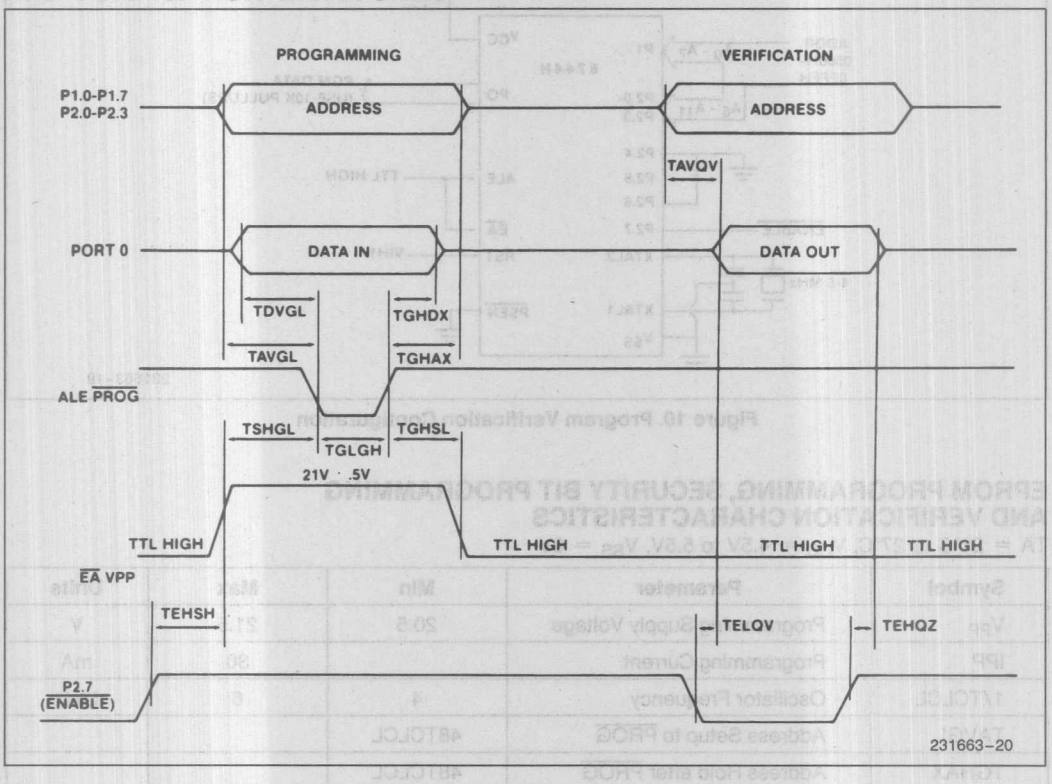
Figure 10. Program Verification Configuration

EPROM PROGRAMMING, SECURITY BIT PROGRAMMING AND VERIFICATION CHARACTERISTICS

TA = 21°C to 27°C, V_{CC} = 4.5V to 5.5V, V_{SS} = 0V

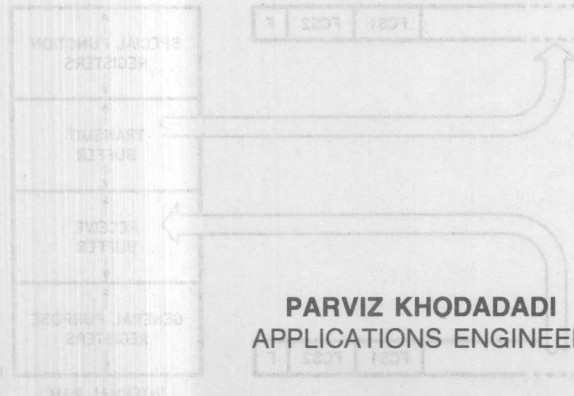
Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	20.5	21.5	V
IPP	Programming Current		30	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{PROG}$	48TCLCL		
TGHAX	Address Hold after $\overline{PROG}$	48TCLCL		
TDVGL	Data Setup to $\overline{PROG}$	48TCLCL		
TGHDX	Data Hold after $\overline{PROG}$	48TCLCL		
TEHSH	$\overline{ENABLE}$ High to Vpp	48TCLCL		
TSHGL	V_{PP} Setup to $\overline{PROG}$	10		µsec
TGHSL	V_{PP} Hold after $\overline{PROG}$	10		µsec
TGLGH	$\overline{PROG}$ Width	45	55	msec
TAVQV	Address to Data Valid		48TCLCL	
TELQV	$\overline{ENABLE}$ to Data Valid		48TCLCL	
TEHQZ	Data Float after $\overline{ENABLE}$	0	48TCLCL	

EPROM PROGRAMMING, SECURITY BIT PROGRAMMING AND VERIFICATION WAVEFORMS



September 1986

Flexibility in Frame Size with the 8044



PARVIZ KHODADADI
APPLICATIONS ENGINEER

Order Number: 292019-001

1.0 INTRODUCTION

The 8044 is a serial communication microcontroller known as the RUPI (Remote Universal Peripheral Interface). It merges the popular 8051 8-bit microcontroller with an intelligent, high performance HDLC/SDLC serial communication controller called the Serial Interface Unit (SIU). The chip provides all features of the microcontroller and supports the Synchronous Data Link Control (SDLC) communications protocol.

There are two methods of operation relating to frame size:

1) Normal operation (limited frame size)
2) Expanded operation (unlimited frame size)

In Normal operation the internal 192 byte RAM is used as the receive and transmit buffer. In this operation, the chip supports data rates up to 2.4 Mbps externally clocked and 375 Kbps self-clocked. For frame sizes greater than 192 bytes, Expanded operation is required. In Expanded operation the external RAM, in conjunction with the internal RAM, is used as the transmit and receive buffer. In this operation, the chip supports data rates up to 500 Kbps externally clocked and 375 Kbps self-clocked. In both cases, the SIU handles many of the data link functions in hardware, and the chip can be configured in either Auto or Flexible mode.

The discussion that follows describes the operation of the chip and the behavior of the serial interface unit. Both Normal and Expanded operations will be further explained with extra emphasis on Expanded operation and its supporting software. Two examples of SDLC communication systems will also be covered, where the chip is used in Expanded operation. The discussion assumes that the reader is familiar with the 8044 data sheet and the SDLC communications protocol.

1.1 Normal Operation

In Normal operation the on-chip CPU and the SIU operate in parallel. The SIU handles the serial communication task while the CPU processes the contents of the on-chip transmit and receiver buffer, services interrupt routines, or performs the local real time processing tasks.

The 192 bytes of on-chip RAM serves as the interface buffer between the CPU and the SIU, used by both as a receive and transmit buffer. Some of the internal RAM space is used as general purpose registers (e.g. R0–R7). The remaining bytes may be divided into at least two sections: one section for the transmit buffer and the other section for the receive buffer. In some applications, the 192 byte internal RAM size imposes a limitation on the size of the information field of each frame and, consequently, achieves less than optimal information throughput.

Figure 1 illustrates the flow of data when internal RAM is used as the receive and transmit buffer. The on-chip CPU allocates a receive buffer in the internal RAM and enables the SIU. A receiving SDLC frame is processed by the SIU and the information bytes of the frame, if any, are stored in the internal RAM. Then, the SIU informs the CPU of the received bytes (Serial Channel interrupt). For transmission, the CPU loads the transmitting bytes into the internal RAM and enables the SIU. The SIU transmits the information bytes in SDLC format.

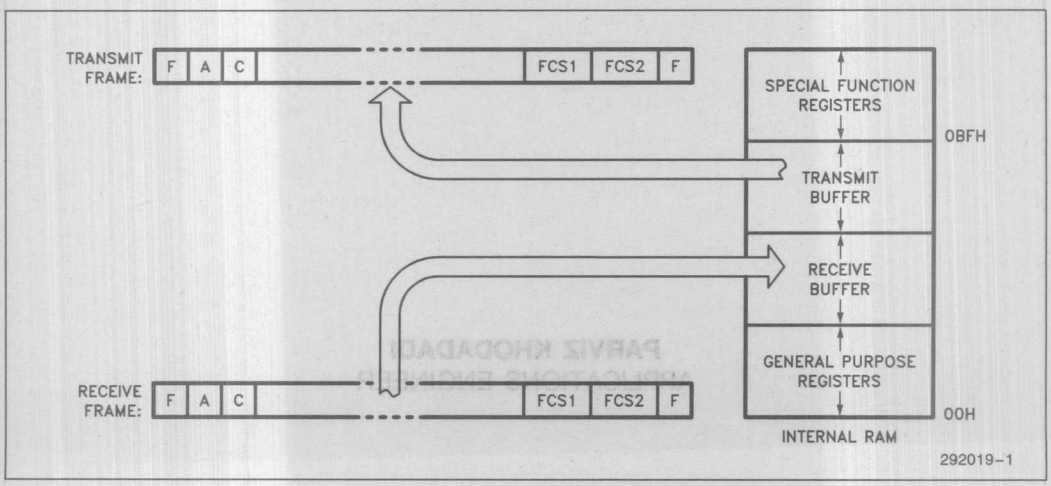


Figure 1. Transmission/Reception Data Flow Using Internal RAM

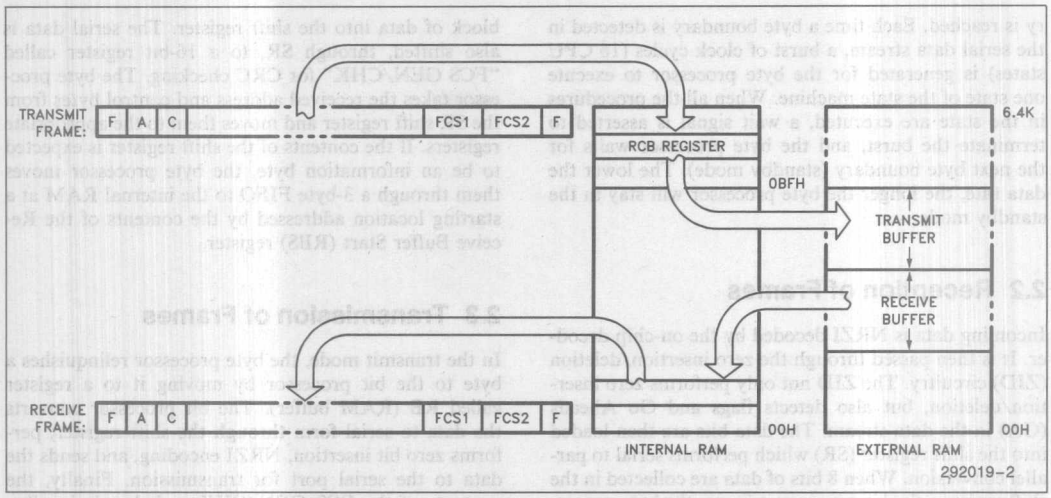


Figure 2. Transmission/Reception Data Flow Using External RAM

1.2 Expanded Operation

In Expanded operation the on-chip CPU monitors the state of the SIU, and moves data from/to external buffer to/from the internal RAM and registers while reception/transmission is taking place. If the CPU must service an interrupt during transmission or reception of a frame or transmit from internal RAM, the chip can shift to Normal operation.

There is a special function register called SIUST, the contents of which dictate the operation of the SIU. Also, at data rates lower than 2.4 Mbps, one section of the SIU, in fixed intervals during transmission and reception, is in the "standby" mode and performs no function. The above two characteristics make it possible to program the CPU to move data to/from external RAM and to force the SIU to repeat or skip some desired hardware tasks while transmission or reception is taking place. With these modifications, external RAM can be utilized as a transmit and receive buffer instead of the internal RAM.

Figure 2 graphically shows the flow of data when external RAM is used. For reception, the receiving bytes are loaded into the Receive Control Byte (RCB) register. Then, the data in RCB is moved to external RAM and the SIU is forced to load the next byte into the RCB register - The chip believes it is receiving a control byte continuously. For transmission, Information bytes (I-bytes) are loaded into a location in the internal RAM and the chip is forced to transmit the contents of this location repeatedly.

Discussion of expanded operation is continued in sections 4 and 5. First, however, sections 2 and 3 describe features of the 8044 which are necessary to further explain expanded operation.

2.0 THE SERIAL INTERFACE UNIT

2.1 Hardware Description

The Serial Interface Unit (SIU) of the RUPI, shown in Figure 3, is divided functionally into a Bit Processor (BIP) and a Byte Processor (BYP), each sharing some common timing and control logic. The bit processor is the interface between the SIU bus and the serial port pins. It performs all functions necessary to transmit/receive a byte of data to/from the serial data line (shifting, NRZI coding, zero insertion/deletion, etc.). The byte processor manipulates bytes of data to perform message formatting, transmitting, and receiving functions. For example, moving bytes from/to the special function registers to/from the bit processor.

The byte processor is controlled by a Finite-State Machine (FSM). For every receiving/transmitting byte, the byte processor executes one state. It then jumps to the next state or repeats the same state. These states will be explained in section 3. The status of the FSM is kept in an 8-bit register called SIUST (SIU State Counter). This register is used to manipulate the behavior of the byte processor.

As the name implies, the bit processor processes data one bit at a time. The speed of the bit processor is a function of the serial channel data rate. When one byte of data is processed by the bit processor, a byte bounda-

ry is reached. Each time a byte boundary is detected in the serial data stream, a burst of clock cycles (16 CPU states) is generated for the byte processor to execute one state of the state machine. When all the procedures in the state are executed, a wait signal is asserted to terminate the burst, and the byte processor waits for the next byte boundary (standby mode). The lower the data rate, the longer the byte processor will stay in the standby mode.

2.2 Reception of Frames

Incoming data is NRZI decoded by the on-chip decoder. It is then passed through the zero insertion/deletion (ZID) circuitry. The ZID not only performs zero insertion/deletion, but also detects flags and Go Aheads (GA) in the data stream. The data bits are then loaded into the shift register (SR) which performs serial to parallel conversion. When 8 bits of data are collected in the shift register, the bit processor triggers the byte processor to process the byte, and it proceeds to load the next block of data into the shift register. The serial data is also shifted, through SR, to a 16-bit register called "FCS GEN/CHK" for CRC checking. The byte processor takes the received address and control bytes from the SR shift register and moves them to the appropriate registers. If the contents of the shift register is expected to be an information byte, the byte processor moves them through a 3-byte FIFO to the internal RAM at a starting location addressed by the contents of the Receive Buffer Start (RBS) register.

2.3 Transmission of Frames

In the transmit mode, the byte processor relinquishes a byte to the bit processor by moving it to a register called RB (RAM buffer). The bit processor converts the data to serial form through the shift register, performs zero bit insertion, NRZI encoding, and sends the data to the serial port for transmission. Finally, the contents of the FCS GEN/CHK and the closing flag are routed to the serial port for transmission.

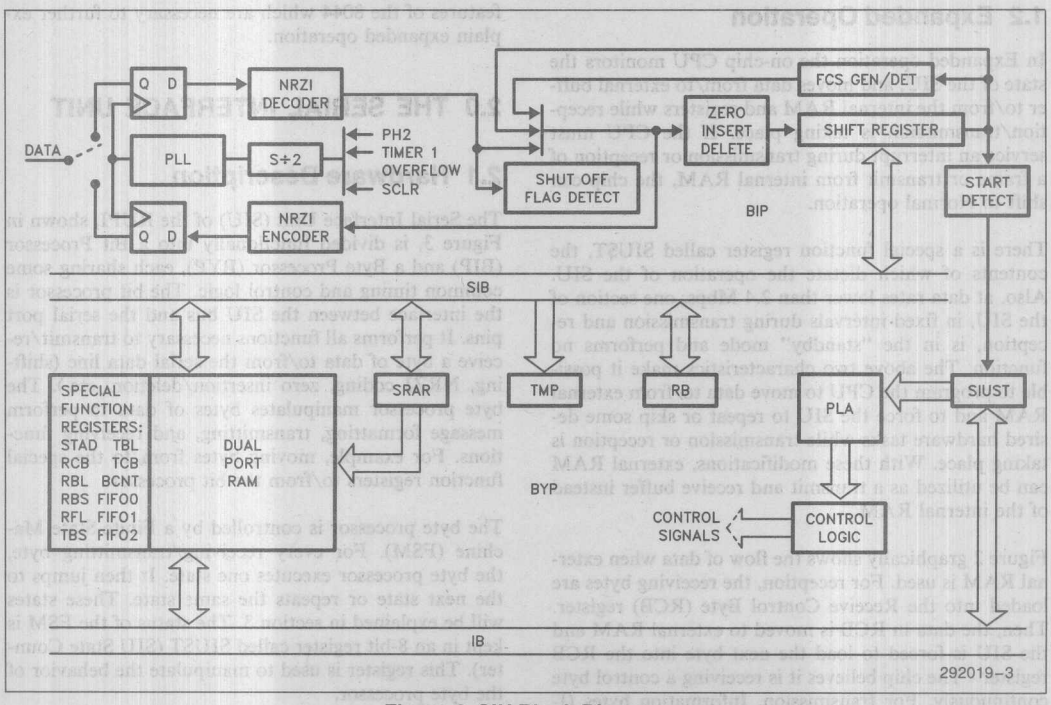


Figure 3. SIU Block Diagram

3.0 TRANSMIT AND RECEIVE STATES

The simplified receive and transmit state diagrams are shown in Figures 4 and 5, respectively. The numbers on the left of each state represent the contents of the SIUST register when the byte processor is in the standby mode, and the instructions on the right of each state represent the "state procedures" of that state. When the byte processor executes these procedures the least three significant bits of the SIUST register are being incremented while the other bits remain unchanged. The byte processor will jump from one state to another without going into the standby mode when a conditional jump procedure executed by the byte processor is true.

3.1 Receive State Sequence

When an opening flag (7EH) is detected by the bit processor, the byte processor is triggered to execute the procedures of the FLAG state. In the FLAG state, the byte processor loads the contents of the RBS register into the Special RAM (SRAR) register. SRAR is the pointer to the internal RAM. The byte processor decrements the contents of the Receive Buffer Length (RBL) register and loads them into the DMA Count (DCNT) register. The FCS GEN/CHK circuit is turned on to monitor the serial data stream for Frame Check Sequence functions as per SDLC specifications.

Assuming there is an address field in the frame, contents of the SIUST register will then be changed to 08H, causing the byte processor to jump to the ADDRESS state and wait (standby) for the next byte boundary. As soon as the bit processor moves the address byte into the SR shift register, a byte boundary is achieved and the byte processor is triggered to execute the procedures in the ADDRESS state.

In the ADDRESS state the received station address is compared to the contents of the STAD register. If there is no match, or the address is not the broadcast address (FFH), reception will be aborted (SIUST = 01H). Otherwise, the byte processor jumps to the CONTROL state (SIUST = 10H) and goes into standby mode.

The byte processor jumps to the CONTROL state if there exists a control field in the receiving frame. In this state the control byte is moved to the RCB register by the byte processor. Note that the only action taken in this state is that a received byte, processed by the bit processor, is moved to RCB. There is no other hardware task performed, and DCNT and SRAR are not affected in this state.

The next two states, PUSH-1 and PUSH-2, will be executed if Frame check sequence (NFCS = 0) option is selected. In these two states the first and second bytes of the information field are pushed into the 3-byte FIFO (FIFO0, FIFO1, FIFO2) and the Receive Field Length register (RFL) is set to zero. The 3-byte FIFO is used as a pipeline to move received bytes into the internal RAM. The FIFO prevents transfer of CRC bytes and the closing flag to the receive buffer (i.e., when the ending flag is received, the contents of FIFO are FLAG, FCS1, and FCS0.) The three byte FIFO is collapsed to one byte in No FCS mode.

In the DMA-LOOP state the byte processor pushes a byte from SR to FIFO0, moves the contents of FIFO2 to the internal RAM addressed by the contents of SRAR, increments the SRAR and RFL registers, and decrements the DCNT register. If more information bytes are expected, the byte processor repeats this state on the next byte boundaries until DMA Buffer End occurs. The DMA Buffer End occurs if SRAR reaches 0BFH (192 decimal), DCNT reaches zero, or the RBP bit of the STS register is set.

The BOV-LOOP state, the last state, is executed if there is a buffer overrun. Buffer overrun occurs when the number of information bytes received is larger than the length of the receive buffer (RFL > RBL). This state is executed until the closing flag is received.

At the end of reception, if the FCS option is used, the closing flag and the FCS bytes will remain in the 3-byte FIFO. The contents of the RCB register are used to update the NSNR (Receive/Send Count) register. The SIU updates the STS register and sets the serial interrupt.

3.2 Transmit State Sequence

Setting the RTS bit puts the SIU in the transmit mode. When the CTS pin goes active, the byte processor goes into START-XMIT state. In this state the opening flag is moved into the RAM Buffer (RB) register. The byte processor jumps to the next state and goes into the standby mode.

If the Pre-Frame Sync (PFS) option is selected, the PFS1 and PFS2 states will be executed to transmit the two Pre-Frame Sync bytes (00H or 55H). In these two states the contents of the Pre-Frame Sync generator are sent to the serial port while the Zero Insertion Circuit (ZID) is turned off. ZID is turned back on automatically on the next byte boundary.

If the PFS option is not chosen, the byte processor jumps to the FLAG state. In this state, the byte processor moves the contents of TBS into the SRAR register, decrements TBL and moves the contents into the DCNT register. The byte processor turns off the ZID and turns on FCS GEN/CHK. The contents of FCS GEN/CHK are not transmitted unless the NFCS bit is

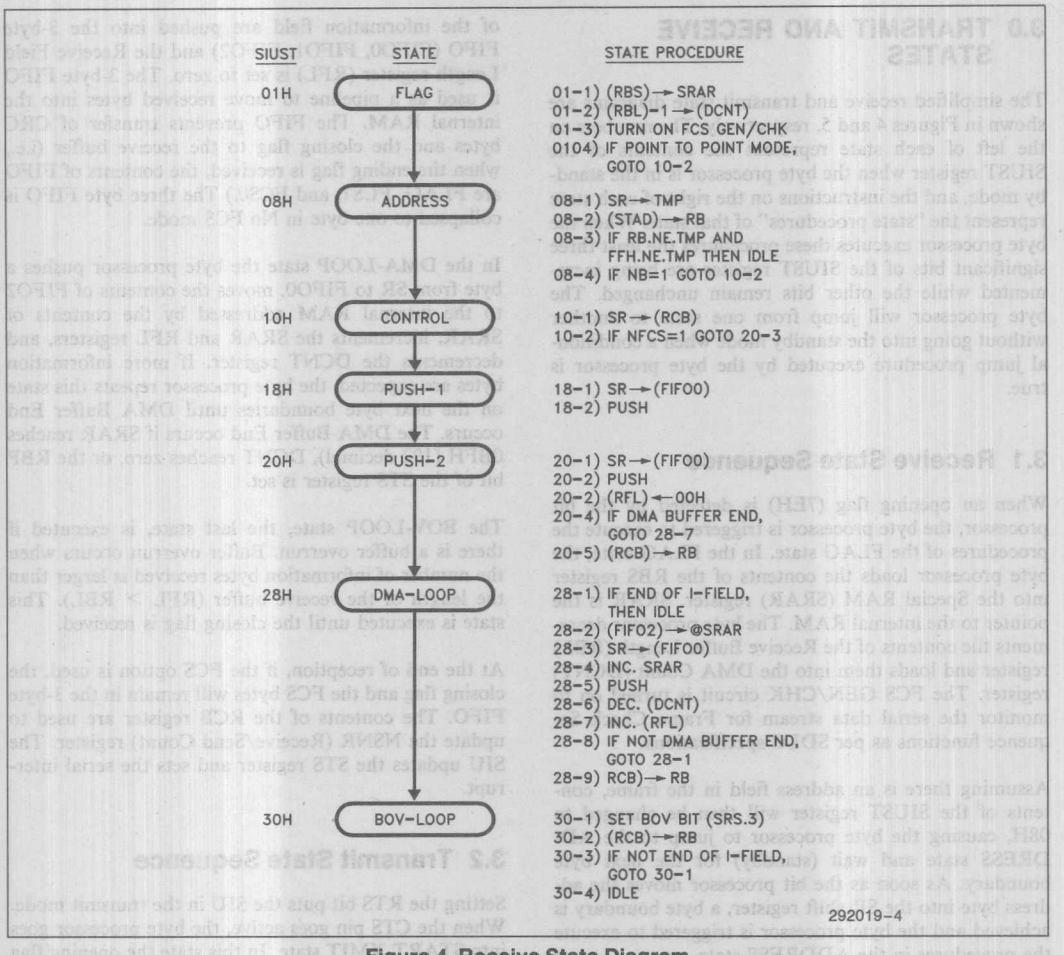


Figure 4. Receive State Diagram

set. If a frame with the address field is chosen, it moves the contents of the STAD register into the RB register for transmission. At the same time, the opening flag is being transmitted by the bit processor.

In the ADDRESS (SIUST = A0H) and CONTROL (SIUST = A8H) states, TCB and the first information byte are loaded into the RB register for transmission, respectively. Note that in the CONTROL state, none of the registers (e.g. DCNT, SRAR) are incremented, and ZID and FCS GEN/CHK are not turned on or off.

The procedures in the DMA-LOOP state are similar to the procedures of the DMA-LOOP in the receive state diagram. The SRAR register pointer to the internal RAM is incremented, and the DCNT register is decremented. The contents of DCNT reach zero when all the information bytes from the transmit buffer are transmitted. A byte from RAM is moved to the RB register for transmission. This state is executed on the following byte boundaries until all the information bytes are transmitted.

The FCS1 and the FCS2 states are executed to transmit the Frame Check Sequence bytes generated by the FCS generator, and the END-FLAG state is executed to transmit the closing flag.

The XMIT-ACTION and the ABORT-ACTION states are executed by the byte processor to synchronize the SIU with the CPU clock. The XMIT-ACTION or the ABORT-ACTION state is repeated until the byte processor status is updated. At the end, the STS and the TMOD registers are updated.

The two ABORT-SEQUENCE states (SIUST = E0H and SIUST = E8H) are executed only if transmission is aborted by the CPU (RTS or TBF bit of the STS register is cleared) or by the serial data link (CTS signal goes inactive or shut-off occurs in loop mode.)

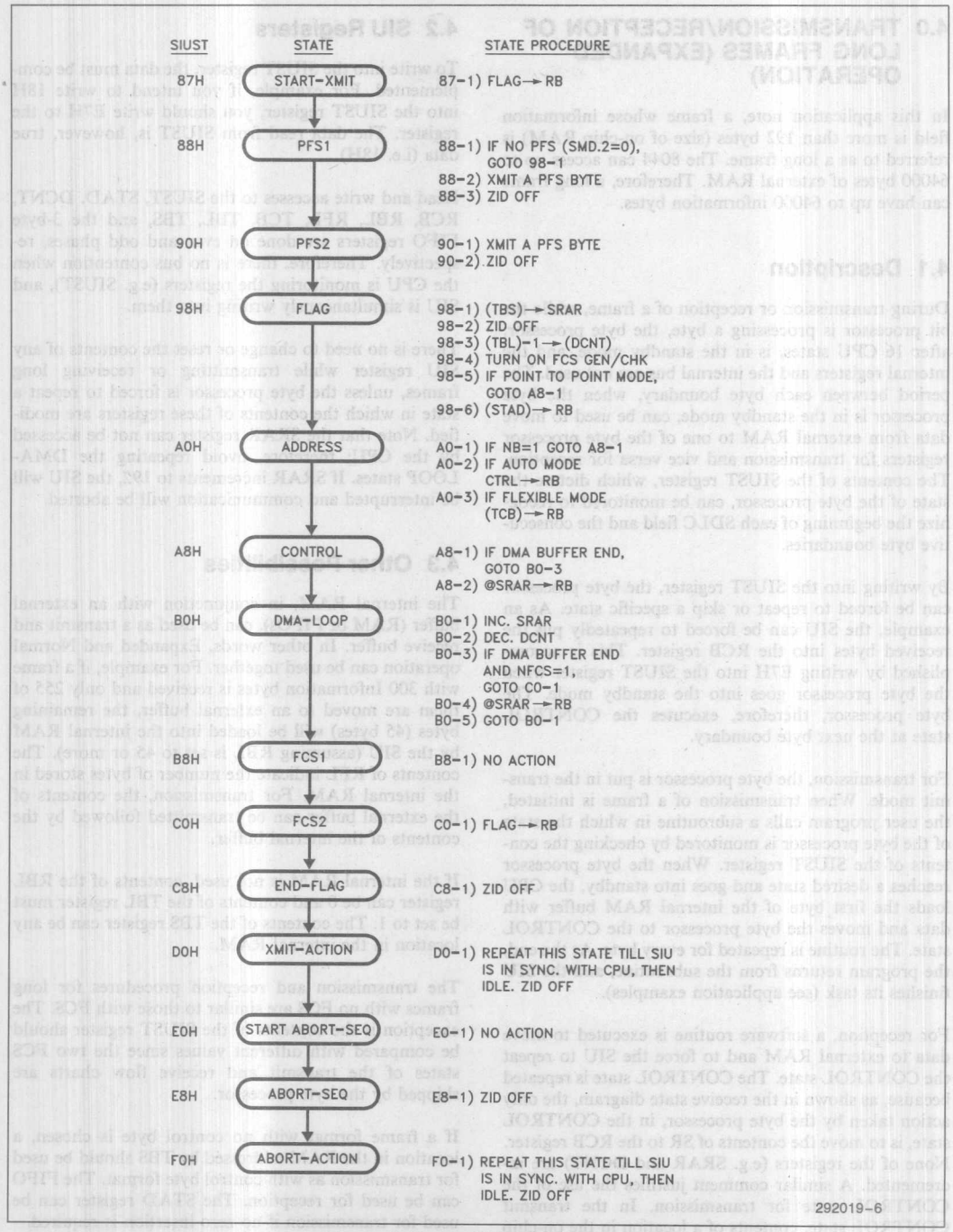


Figure 5. Transmit State Diagram

4.0 TRANSMISSION/RECEPTION OF LONG FRAMES (EXPANDED OPERATION)

In this application note, a frame whose information field is more than 192 bytes (size of on-chip RAM) is referred to as a long frame. The 8044 can access up to 64000 bytes of external RAM. Therefore, a long frame can have up to 64000 information bytes.

4.1 Description

During transmission or reception of a frame, while the bit processor is processing a byte, the byte processor, after 16 CPU states, is in the standby mode, and the internal registers and the internal bus are not used. The period between each byte boundary, when the byte processor is in the standby mode, can be used to move data from external RAM to one of the byte processor registers for transmission and vice versa for reception. The contents of the SIUST register, which dictate the state of the byte processor, can be monitored to recognize the beginning of each SDLC field and the consecutive byte boundaries.

By writing into the SIUST register, the byte processor can be forced to repeat or skip a specific state. As an example, the SIU can be forced to repeatedly put the received bytes into the RCB register. This is accomplished by writing E7H into the SIUST register when the byte processor goes into the standby mode. The byte processor, therefore, executes the CONTROL state at the next byte boundary.

For transmission, the byte processor is put in the transmit mode. When transmission of a frame is initiated, the user program calls a subroutine in which the state of the byte processor is monitored by checking the contents of the SIUST register. When the byte processor reaches a desired state and goes into standby, the CPU loads the first byte of the internal RAM buffer with data and moves the byte processor to the CONTROL state. The routine is repeated for every byte. At the end, the program returns from the subroutine, and the SIU finishes its task (see application examples).

For reception, a software routine is executed to move data to external RAM and to force the SIU to repeat the CONTROL state. The CONTROL state is repeated because, as shown in the receive state diagram, the only action taken by the byte processor, in the CONTROL state, is to move the contents of SR to the RCB register. None of the registers (e.g. SRAR and DCNT) are incremented. A similar comment justifies the use of the CONTROL state for transmission. In the transmit CONTROL state, contents of a location in the on-chip RAM addressed by TBS is moved to RB for transmission.

4.2 SIU Registers

To write into the SIUST register, the data must be complemented. For example, if you intend to write 18H into the SIUST register, you should write E7H to the register. The data read from SIUST is, however, true data (i.e. 18H).

Read and write accesses to the SIUST, STAD, DCNT, RCB, RBL, RFL, TCB, TBL, TBS, and the 3-byte FIFO registers are done on even and odd phases, respectively. Therefore, there is no bus contention when the CPU is monitoring the registers (e.g. SIUST), and SIU is simultaneously writing into them.

There is no need to change or reset the contents of any SIU register while transmitting or receiving long frames, unless the byte processor is forced to repeat a state in which the contents of these registers are modified. Note that the SRAR register can not be accessed by the CPU; therefore, avoid repeating the DMA-LOOP states. If SRAR increments to 192, the SIU will be interrupted and communication will be aborted.

4.3 Other Possibilities

The internal RAM, in conjunction with an external buffer (RAM or FIFOs), can be used as a transmit and receive buffer. In other words, Expanded and Normal operation can be used together. For example, if a frame with 300 Information bytes is received and only 255 of them are moved to an external buffer, the remaining bytes (45 bytes) will be loaded into the internal RAM by the SIU (assuming RBL is set to 45 or more). The contents of RFL indicate the number of bytes stored in the internal RAM. For transmission, the contents of the external buffer can be transmitted followed by the contents of the internal buffer.

If the internal RAM is not used, contents of the RBL register can be 0 and contents of the TBL register must be set to 1. The contents of the TBS register can be any location in the internal RAM.

The transmission and reception procedures for long frames with no FCS are similar to those with FCS. The exception is the contents of the SIUST register should be compared with different values since the two FCS states of the transmit and receive flow charts are skipped by the byte processor.

If a frame format with no control byte is chosen, a location in the RAM addressed by TBS should be used for transmission as with control byte format. The FIFO can be used for reception. The STAD register can be used for transmission if no zero insertion is required.

If the RUPI is used in Auto mode (see Section 5), it will still respond to RR, RNR, REJ, and Unnumbered Poll (UP) SDLC commands with RR or RNR automatically, without using any transmit routine. For example, if the on-chip CPU is busy performing some real time operations, the SIU can transmit an information frame from the internal buffer or transmit a supervisory frame without the help of CPU (Normal operation).

Maximum data rate using this feature is limited primarily by the number of instructions needed to be executed during the standby mode.

Transmission or reception of a frame can be timed out so that the CPU will not hang up in the transmit or receive procedures if a frame is aborted. Or, if the data rate allows enough time (standby time is long enough), the CPU can monitor the SIUST register for idle mode (SIUST = 01H).

It is also possible to transmit multiple opening or closing flags by forcing the byte processor to repeat the END-FLAG state.

4.4 Maximum Data Rate in Expanded Operation

Assuming there is no zero-insertion/deletion, the bit processor requires eight serial clock periods to process one block of data. The byte processor, running on the CPU clock, processes one byte of data in 16 CPU states (one state of the state diagrams). Each CPU state is two oscillator periods. At an oscillator frequency of 12 MHz, the CPU clock is 6 MHz, and 16 CPU states is 2.7 μs. At a 3 Mbit rate with no zero-insertion/deletion, there is exactly enough time to execute one state per byte (16 states at 6 MHz = 8 bits at 3M baud). In other words, the standby time is zero.

Figure 6 demonstrates portions of the timing relationship between the byte processor and the bit processor. In each state, the actions taken by the processors, plus the contents of the SIUST register, are shown. When the byte processor is running, the contents of SIUST are unknown. However, when it is in the standby mode, its contents are determinable.

The maximum data rate for transmitting and receiving long frames depends on the number of instructions needed to be executed during standby, and is propor-

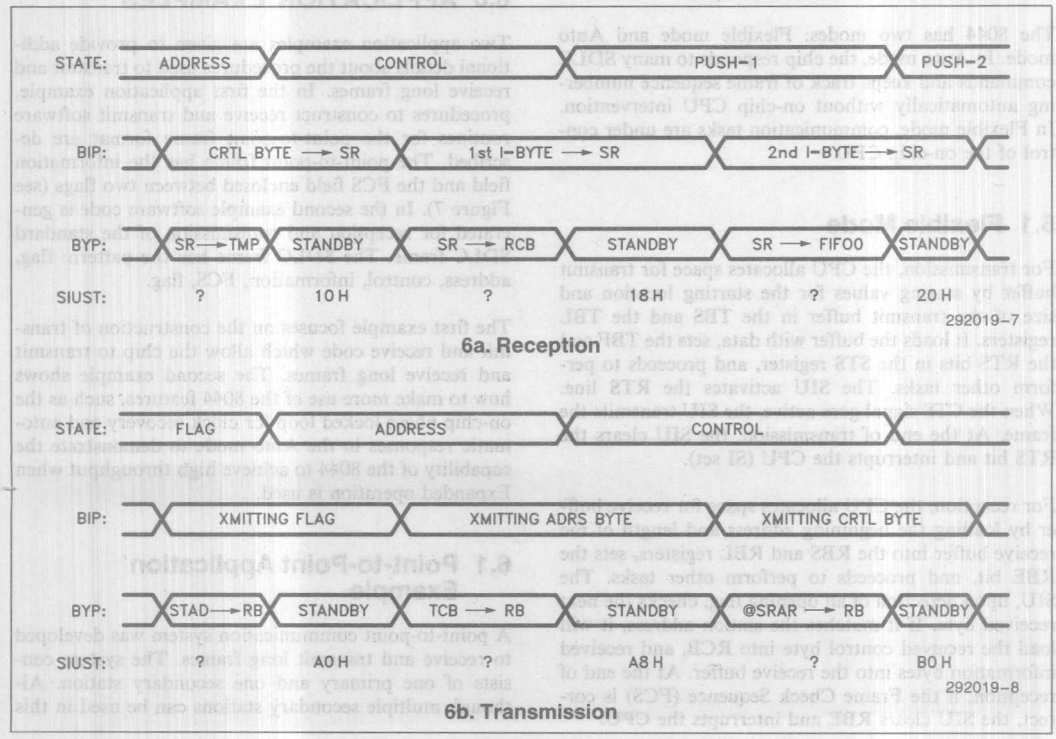


Figure 6. Portions of the BIP/BYP Timing Relationship

tional to the oscillator frequency. The time the byte processor is in the standby mode, waiting for the bit processor to deliver a processed byte, is at least equal to eight serial clock periods minus 16 CPU states. If an inserted zero is in the block of data, the bit processor will process the byte in nine serial clock periods.

The equation for theoretical maximum data rate is given as:

$$(2TCLCL) \times (16 \text{ states}) + (\#\text{ of instruction cycles}) \times (12TCLCL) = (8TDCY) \quad \text{Equation (1)}$$

Where: TCLCL is the oscillator period.
TDCY is the serial clock period.

At an oscillator frequency of 12 MHz and baud rate of 375 Kbps, about 18 instruction cycles can be executed when the byte processor is in the standby mode. At a 9600 baud rate, there is time to execute about 830 instruction cycles—plenty of time to service a long interrupt routine or perform bit-manipulation or arithmetic operations on the data while transmission or reception is taking place.

5.0 MODES OF OPERATION

The 8044 has two modes: Flexible mode and Auto mode. In Auto mode, the chip responds to many SDLC commands and keeps track of frame sequence numbering automatically without on-chip CPU intervention. In Flexible mode, communication tasks are under control of the on-chip CPU.

5.1 Flexible Mode

For transmission, the CPU allocates space for transmit buffer by storing values for the starting location and size of the transmit buffer in the TBS and the TBL registers. It loads the buffer with data, sets the TBF and the RTS bits in the STS register, and proceeds to perform other tasks. The SIU activates the RTS line. When the CTS signal goes active, the SIU transmits the frame. At the end of transmission, the SIU clears the RTS bit and interrupts the CPU (SI set).

For reception, the CPU allocates space for receive buffer by loading the beginning address and length of the receive buffer into the RBS and RBL registers, sets the RBE bit, and proceeds to perform other tasks. The SIU, upon detection of an opening flag, checks the next received byte. If it matches the station address, it will load the received control byte into RCB, and received information bytes into the receive buffer. At the end of reception, if the Frame Check Sequence (FCS) is correct, the SIU clears RBE and interrupts the CPU.

5.2 Auto Mode

In the Auto mode, the 8044 can only be a secondary station operating in the SDLC "Normal Response Mode". The 8044 in Auto mode does not transmit messages unless it is polled by the primary.

For transmission of an information frame, the CPU allocates space for the transmit buffer, loads the buffer with data, and sets the TBF bit. The SIU will transmit the frame when it receives a valid poll-frame. A frame whose poll bit of the control byte is set, is a poll-frame. The poll bit causes the RTS bit to be set. If TBF were not set, the SIU would respond with Receive Not Ready (RNR) SDLC command if RBP = 1, or with Receive Ready (RR) SDLC command if RBP = 0. After transmission RTS is cleared, and the CPU is not interrupted.

For reception, the procedure is the same as that of Flexible mode. In addition, the SIU sets the RTS bit if the received frame is a poll-frame (causing an automatic response) and increments the NS and NR counts accordingly.

6.0 APPLICATION EXAMPLES

Two application examples are given to provide additional details about the procedures used to transmit and receive long frames. In the first application example, procedures to construct receive and transmit software routines for the point-to-point frame format are described. The point-to-point frame has the information field and the FCS field enclosed between two flags (see Figure 7). In the second example software code is generated for reception and transmission of the standard SDLC frame. The SDLC frame has the pattern: flag, address, control, information, FCS, flag.

The first example focuses on the construction of transmit and receive code which allow the chip to transmit and receive long frames. The second example shows how to make more use of the 8044 features, such as the on-chip phase locked loop for clock recovery and automatic responses in the Auto mode to demonstrate the capability of the 8044 to achieve high throughput when Expanded operation is used.

6.1 Point-to-Point Application Example

A point-to-point communication system was developed to receive and transmit long frames. The system consists of one primary and one secondary station. Although multiple secondary stations can be used in this

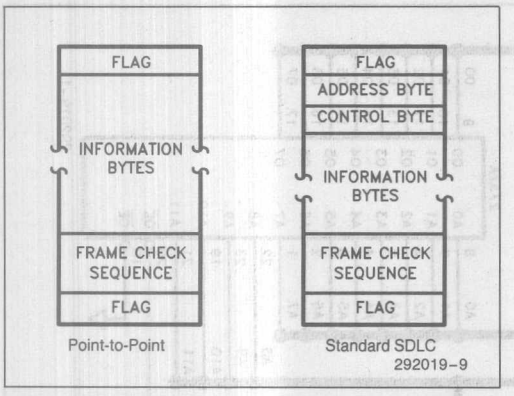


Figure 7. Point-to-Point and Standard SDLC Frame Formats

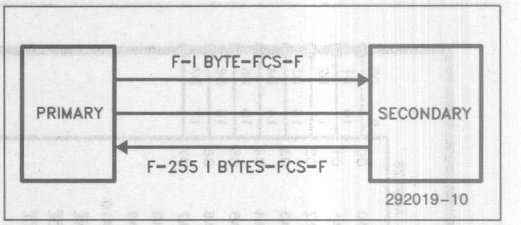


Figure 8. Secondary Responses to Primary Station Commands

system, one secondary is chosen to simplify the primary station's software and focus on the long frame software code. Both the primary and the secondary stations are in Flexible mode and the external clock option is used for the serial channel. The maximum data rate is 500 Kbps. The FCS bytes are generated and checked automatically by both stations.

6.1.1 POLLING SEQUENCE

The polling sequence, shown in Figure 8, takes place continuously between the primary and the secondary stations. The primary transmits a frame with one information byte to the secondary. The information byte is used by the secondary as an address byte. The secondary checks the received byte, and if the address matches, the secondary responds with a long frame. In this example, the information field of the frame is chosen to be 255 bytes long. Since there is only one secondary station, the address always matches. Upon successful reception of the long frame, the primary transmits another frame to the secondary station.

6.1.2 HARDWARE

The schematic of the secondary station is given in Figure 9. The circuit of the primary station is identical to the secondary station with the exception of pin 11 (DATA) being connected to pin 14 (T0). In the primary station, the 8044 is interrupted when activity is detected on the communication line by the on-chip timer (in counter mode). This is explained more later. The serial clock to both stations is supplied by a pulse generator. The output of the pulse generator (not shown in the diagram) is connected to pin 15 of the 8044s. Since the two stations are located near each other (less than 4 feet), line drivers are not used.

The central processor of each station is the 8044. The data link program is stored in a 2Kx8 EPROM (2732A), and a 2Kx8 static RAM (AM9128) is used as the external transmit and receive buffer. The RTS pin is connected to the CTS pin. For simplicity, the stations are assumed to be in the SDLC Normal Respond Mode after Hardware reset.

6.1.3 PRIMARY STATION SOFTWARE

The assembly code for the primary station software is listed in Appendix A. The primary software consists of the main routine, the SIU interrupt routine, and the receive interrupt routine. The receive interrupt routine is executed when a long frame is being received.

In the flow charts that follow, all actions taken by the SIU appear in squares, and actions taken by the on-chip CPU appear in spheres.

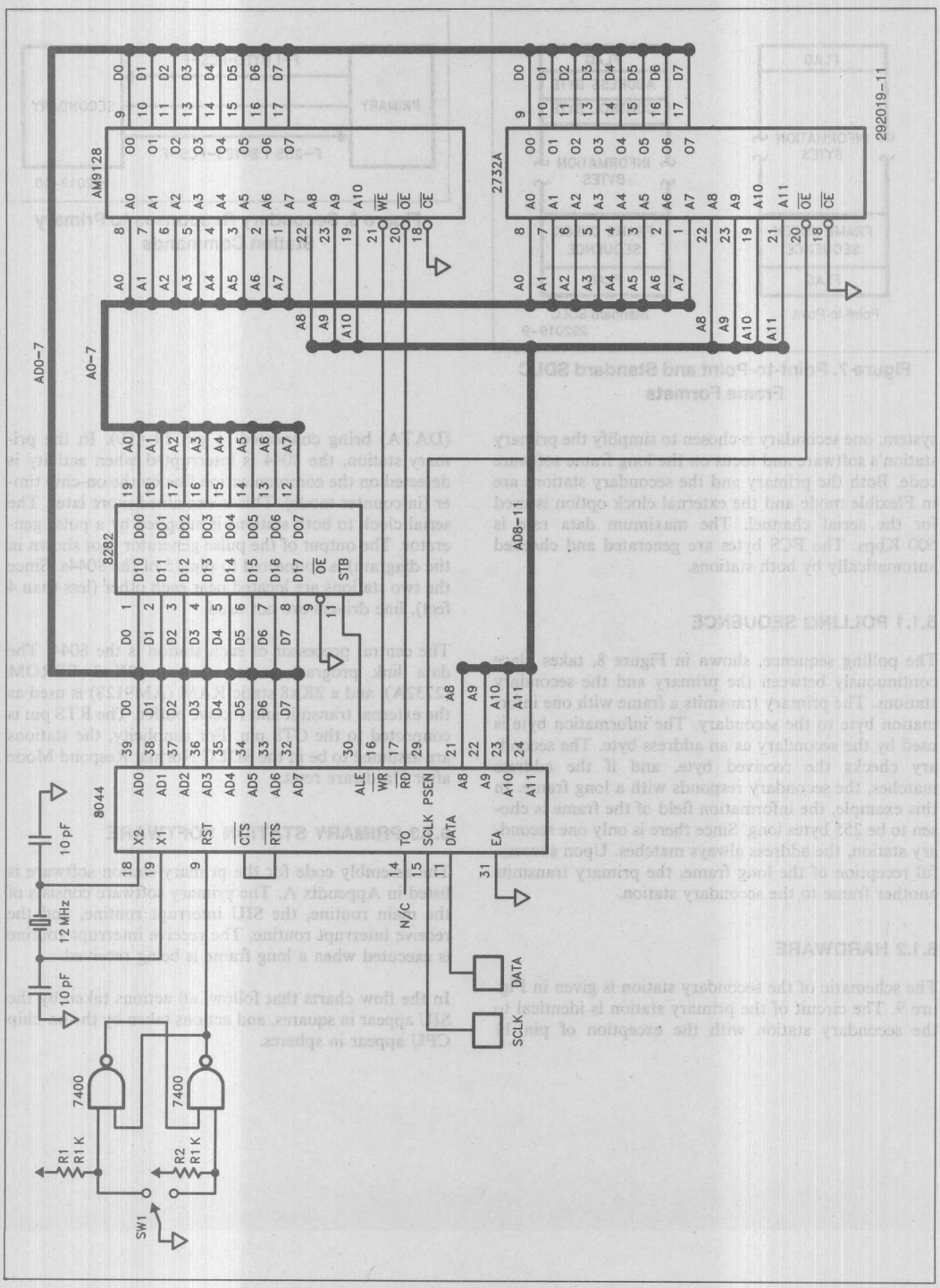


Figure 9. Secondary Station Hardware

Main Routine

First, the chip is initialized (see Figure 10). It is put in Flexible mode, externally clocked, and "Flag-Information Field-FCS-Flag" frame format. Pre-Frame Sync option (PFS = 1) and automatic Frame Check Sequence generation/detection (NFCS = 0) are selected. The on-chip transmit buffer starts at location 20H and the transmit buffer length is set to 1. This one byte buffer contains the address of the secondary station. There is no on-chip receive buffer since the long frame being received is moved to the external buffer. The RTS, TBF, and RBE bits are set simultaneously. Setting the RTS and TBF bits causes the SIU to transmit the contents of the transmit buffer.

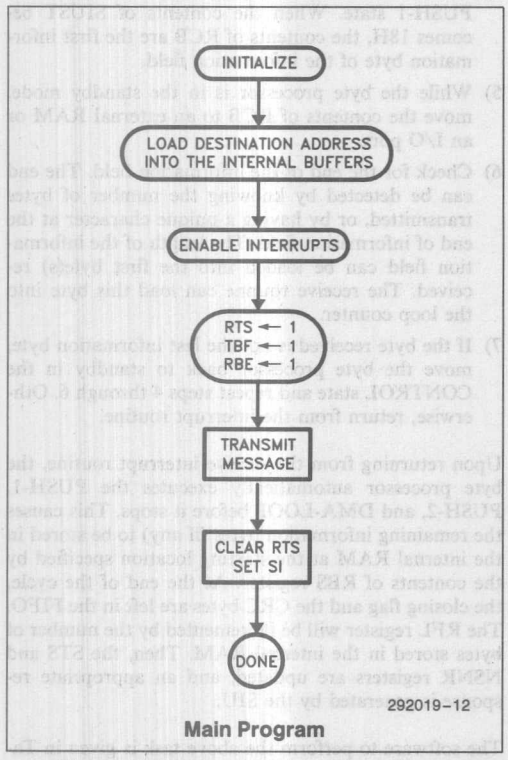


Figure 10. Primary Station Flow Charts

SIU Interrupt Routine

After transmission of the frame, the SIU interrupts the on-chip CPU (SI is set). In the SIU interrupt service routine, counter 0 is initialized and turned on (see Figure 11). The user program returns to perform other tasks. After reception of the long frame, the SIU interrupt routine is executed again. This time, RTS, TBF, and RBE are set for another round of information exchange between the two stations.

SIU never interrupts while reception or transmission is taking place. The SIU registers are updated and the SI is set (serial interrupt) after the closing flag has been received or transmitted. An SIU interrupt never occurs if the receive interrupt routine or the transmit subroutine is being executed.

Setting the RBE bit of the STS register puts the RUPI in the receive mode. However, the jump to the receive interrupt routine occurs only when a frame appears on the serial port. Incoming frames can be detected using the Pre-Frame Sync. option and one of the CPU timers in counter mode. The counter external pin (T0) is connected to the data line (pin 11 is tied to pin 14). Setting the PFS (Pre-Frame Sync.) bit will guarantee 16 transitions before the opening flag of a flame.

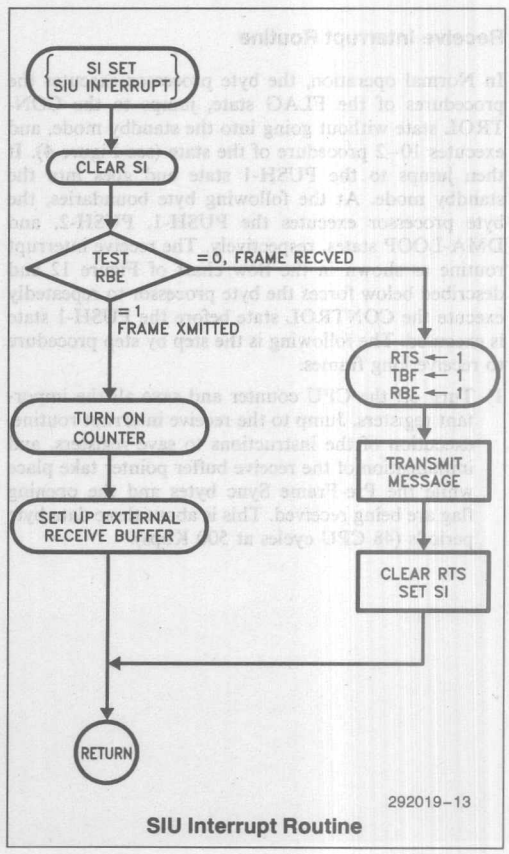


Figure 11. Primary Station Flow Charts

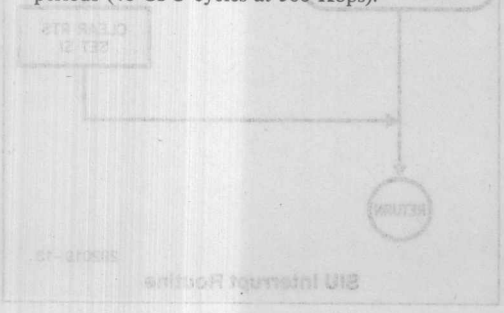
The counter registers are initialized such that the counter interrupt occurs before the opening flag of a frame. When the PFS transitions appear on the data line, the counter overflows and interrupts the CPU. The CPU program jumps to the timer interrupt service routine and executes the receive routine. In the receive routine, the received frame is processed, and the information bytes are moved to the external RAM. Note that the maximum count rate of the 8051 counter is $1/24$ of the oscillator frequency. At 12 MHz, the data rate is limited to 500 Kbps.

Another method to detect a frame on the data line and cause an interrupt is to use an external "Flag-Detect" circuit to interrupt the CPU. The "Flag Detect" circuit can be an 8-bit shift register plus some TTL chips. If this option is used, the RUPI must operate in externally clocked mode because the clock is needed to shift the incoming data into the shift register. With this option, the maximum data rate is not limited by the maximum count rate of the 8051 counter.

Receive Interrupt Routine

In Normal operation, the byte processor executes the procedures of the FLAG state, jumps to the CONTROL state without going into the standby mode, and executes 10–2 procedure of the state (see Figure 4). It then jumps to the PUSH-1 state and goes into the standby mode. At the following byte boundaries, the byte processor executes the PUSH-1, PUSH-2, and DMA-LOOP states, respectively. The receive interrupt routine as shown in the flow chart of Figure 12 and described below forces the byte processor to repeatedly execute the CONTROL state before the PUSH-1 state is executed. The following is the step by step procedure to receive long frames:

1) Turn off the CPU counter and save all the important registers. Jump to the receive interrupt routine, execution of the instructions to save registers, and initialization of the receive buffer pointer take place while the Pre-Frame Sync bytes and the opening flag are being received. This is about three data byte periods (48 CPU cycles at 500 Kbps).



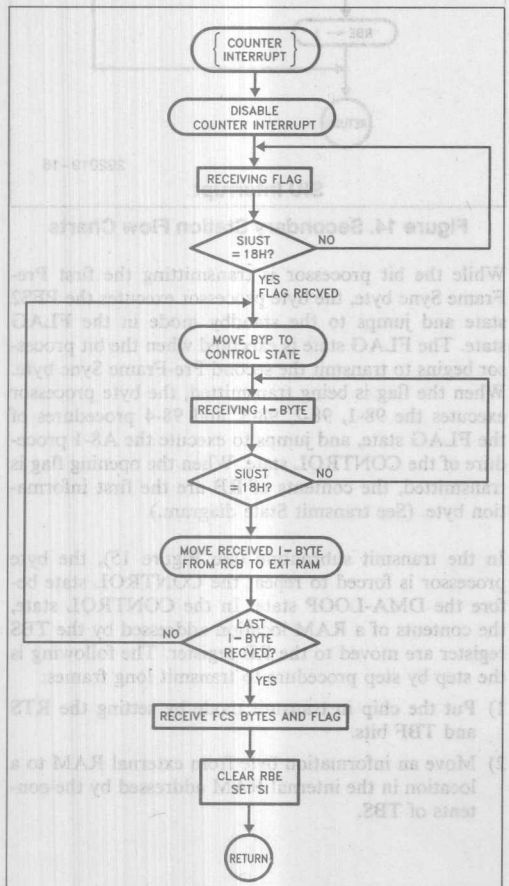
2) Monitor the SIUST register for standby in the PUSH-1 state (SIUST = 18H). When the SIUST contents are 18H, the byte processor is waiting for the first information byte. The bit processor has already recognized the flag and is processing the first information byte.
3) In the standby mode, move the byte processor into the CONTROL state by writing "EFH" (complement of 10H) into the SIUST register. When the next byte boundary occurs, the bit processor has processed and moved a byte of data into the SR register. The byte processor moves the contents of SR into the RCB register, jumps to the PUSH-1 state (SIUST = 18H), and waits.
4) Monitor the SIUST register for standby in the PUSH-1 state. When the contents of SIUST becomes 18H, the contents of RCB are the first information byte of the information field.
5) While the byte processor is in the standby mode, move the contents of RCB to an external RAM or an I/O port.
6) Check for the end of the information field. The end can be detected by knowing the number of bytes transmitted, or by having a unique character at the end of information field. The length of the information field can be loaded into the first byte(s) received. The receive routine can load this byte into the loop counter.
7) If the byte received is not the last information byte, move the byte processor back to standby in the CONTROL state and repeat steps 4 through 6. Otherwise, return from the interrupt routine.

Upon returning from the receive interrupt routine, the byte processor automatically executes the PUSH-1, PUSH-2, and DMA-LOOP before it stops. This causes the remaining information bytes (if any) to be stored in the internal RAM at the starting location specified by the contents of RBS register. At the end of the cycle, the closing flag and the CRC bytes are left in the FIFO. The RFL register will be incremented by the number of bytes stored in the internal RAM. Then, the STS and NSNR registers are updated, and an appropriate response is generated by the SIU.

The software to perform the above task is given in Table 1. In this example, the number of instruction cycles executed during standby is 12 cycles.

Table 1. Codes for Long Frame Reception

```
Receive Codes                                              Cycles
                    .          .
                    .          .
                    .          .
REC:        CLR     TR0
            MOV     A,#18H
WAIT1:      CJNE    A,SIUST,WAIT1
NEXTI:      MOV     SIUST,#0EFH ..........................2
            MOV     A,#18H ...............................1
WAIT2:      CJNE    A,SIUST,WAIT2.........................2
            MOV     A,RCB.................................1
            MOVX    @DPTR,A ..............................2
            INC     DPTR..................................2
            DJNZ    R5,NEXTI..............................2
            RETI
END                                                        12 Cycles
```



Receive Interrupt Routine

Figure 12. Primary Station Flow Charts

6.1.4 SECONDARY STATION SOFTWARE

The assembly code for the secondary station software is given in Appendix A. The secondary station contains the transmit subroutine which is called for transmission of long frames.

Main Routine

As shown in the secondary station flow chart (Figure 13), the external transmit buffer (external RAM) is loaded with the information data (FFH, FEH, FDH, . . .) at starting location 200H. The internal transmit buffer (on chip RAM) starts at location 20H (TBS = 20H), and the transmit buffer length (TBL) is set to 1. The on-chip CPU, in the transmit subroutine, moves the information bytes from the external RAM to this one byte buffer for transmission. The receive buffer starts at location 10H and the receiver buffer length is 1. This buffer is used to buffer the frame transmitted by the primary. The received byte is used as an address byte.

The Secondary is configured like the Primary station. It is put in Flexible mode, externally clocked, Point-to-point frame format. The PFS bit is set to transmit two bytes before the first flag of a frame. The RBE bit is set to put the chip in receive mode. Upon reception of a valid frame, the SIU loads the received information byte into the on-chip receive buffer and interrupts the CPU.

SIU Interrupt Routine

In the serial interrupt routine, the RBE bit is checked (see Figure 14). Since RBE is clear, a frame has been received. The received Information byte is compared with the contents of the Station Address (STAD) register.

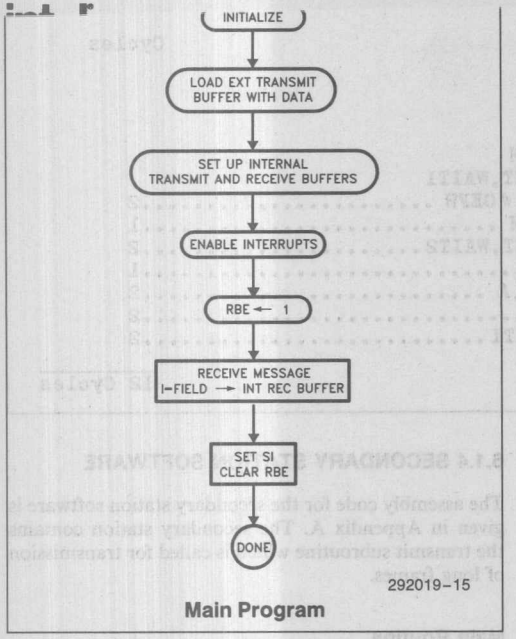


Figure 13. Secondary Station Flow Charts

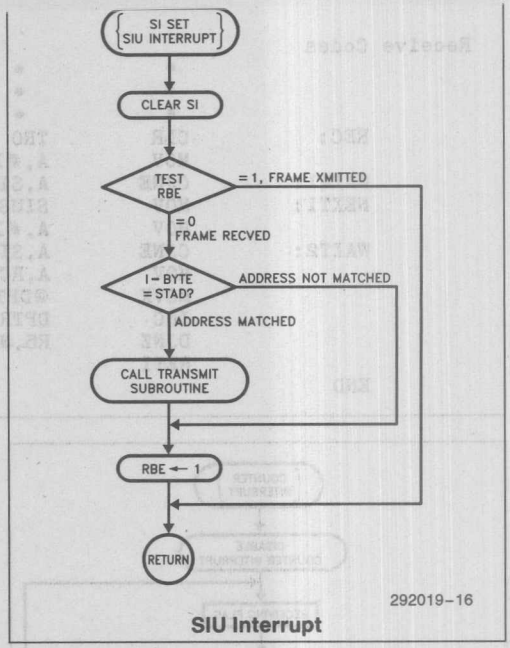


Figure 14. Secondary Station Flow Charts

If they match, the secondary will call the transmit subroutine to transmit the long frame. Upon returning from the transmit subroutine, the RBE bit is set, and program returns from the SIU interrupt. After transmission of the closing flag, SIU interrupt occurs again. In the interrupt routine, the RBE is checked. Since the RBE is set, the program returns from the SIU interrupt routine and waits until another long frame is received.

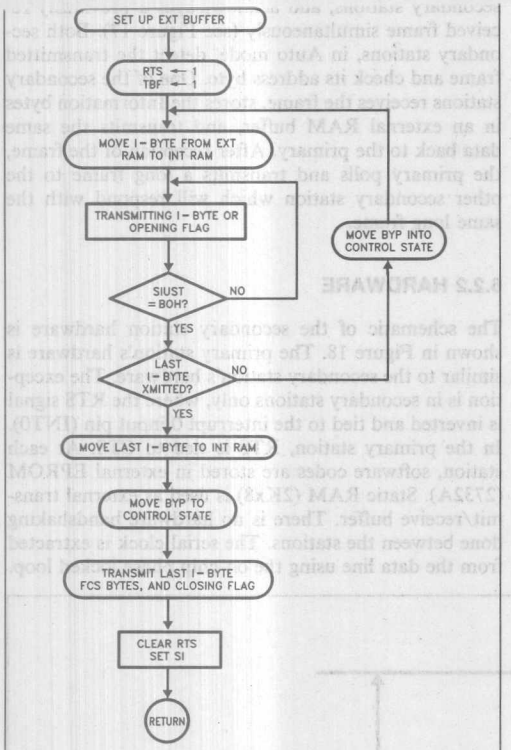
If the secondary were in Auto mode, the chip must be ready to execute the transmit routine upon reception of a poll-frame; otherwise, the chip automatically transmits the contents of the internal transmit buffer if the TBF bit is set, or transmits a supervisory command (RR or RNR) if TBF is clear.

Transmit Subroutine

In Normal operation the byte processor executes the START-TRANSMIT state and jumps to the PFS1 state. While the bit processor is transmitting some unwanted bits, the byte processor executes the PFS1 state and jumps to the standby mode in the PFS2 state. While the bit processor is transmitting the first Pre-Frame Sync byte, the byte processor executes the PFS2 state and jumps to the standby mode in the FLAG state. The FLAG state is executed when the bit processor begins to transmit the second Pre-Frame Sync byte. When the flag is being transmitted, the byte processor executes the 98-1, 98-2, 98-3, and 98-4 procedures of the FLAG state, and jumps to execute the A8-1 procedure of the CONTROL state. When the opening flag is transmitted, the contents of RB are the first information byte. (See transmit State diagram.)

In the transmit subroutine (see Figure 15), the byte processor is forced to repeat the CONTROL state before the DMA-LOOP state. In the CONTROL state, the contents of a RAM location addressed by the TBS register are moved to the RB register. The following is the step by step procedure to transmit long frames:

1) Put the chip in transmit mode by setting the RTS and TBF bits.
2) Move an information byte from external RAM to a location in the internal RAM addressed by the contents of TBS.



Transmit Subroutine

Figure 15. Secondary Station Flow Charts

3) Monitor the SIUST register for the standby mode in the DMA-LOOP state (SIUST = B0H). When SIUST is B0H, the opening flag has been transmitted, and the first information byte is being transmitted by the bit processor.
4) If there are more information bytes, move the byte processor back to the CONTROL state, and repeat steps 2 through 4. Otherwise, continue.
5) Move byte processor to the Standby mode in the CONTROL state (SIUST = A8H) and return from the subroutine.

The byte processor automatically executes the remaining states to send the FCS bytes and the closing flag. After the completion of transmission, SIU updates the STS and NSNR registers and interrupts the CPU.

If the contents of the TBL register were more than 1, the SIU transmits (TBL)−1 additional bytes from the internal RAM at starting address (TBS)+1 because it executes the DMA-LOOP state (TBL)−1 additional times. The byte processor should not be programmed to skip the DMA-LOOP state, because the transmission of FCS bytes is enabled in this state.

The maximum baud rate that can be used with these codes is calculated by adding the number of instruction cycles executed, during the standby mode, between each byte boundaries (see Table 2).

Using Equation 1, the maximum data rate, based on the transmit software, is 509 Kbps; However, the maximum count rate of the counter limits the data rate to 500 Kbps.

Table 2. Codes for Long Frame Transmission

```
Transmit Codes                                              Cycles
                    •       •
                    •       •
                    •       •
TRAN:               MOV     DPTR,#200H
                    MOV     R5,#0FFH
                    SETB    TBF
                    SETB    RTS
LOOP:               MOVX    A,@DPTR
                    MOV     @R1,A
                    MOV     A,#0B0H
WAIT1:              CJNE    A,SIUST,WAIT1.....................2
                    INC     DPTR..............................2
                    MOVX    A,@DPTR ..........................2
                    MOV     @R1,A ............................1
                    DJNZ    R5,NEXTI..........................2
                    MOV     SIUST,#57H
                    RET
NEXTI:              MOV     SIUST,#57H .......................2
                    MOV     A,#0B0H ..........................1
                    JMP     WAIT1.............................1
END
                                                            13 Cycles
```

6.2 Multidrop Application

Performance of long frame in addition to the features of the 8044 are described using a simple multidrop communication system in which three RUPIs, one as a master and the other two as secondary stations, transmit and receive long frames alternately (see Figure 16). All stations perform automatic zero bit insertion/deletion, NRZI decoding/encoding, Frame Check Sequence (FCS) generation/detection, and on-chip clock recovery at a data rate of 375 Kbps.

The primary and the secondary station's software code is given in Appendix B. These programs, for simplicity, assume only reception of information and supervisory frames. It is also assumed that the frames are received and transmitted in order. All stations use very similar transmit and receive routines. This code is written for standard SDLC frames (see Figure 7).

6.2.1 POLLING SEQUENCE

The primary station, in Flexible mode, transmits a long frame (for this example, 255 I-bytes), polls one of the secondary stations, and acknowledges a previously received frame simultaneously (see Figure 17). Both secondary stations, in Auto mode, detect the transmitted frame and check its address byte. One of the secondary stations receives the frame, stores the Information bytes in an external RAM buffer, and transmits the same data back to the primary. After reception of the frame, the primary polls and transmits a long frame to the other secondary station which will respond with the same long frame.

6.2.2 HARDWARE

The schematic of the secondary station hardware is shown in Figure 18. The primary station's hardware is similar to the secondary station's hardware. The exception is in secondary stations only, where the RTS signal is inverted and tied to the interrupt 0 input pin (INT0). In the primary station, RTS is tied to CTS. At each station, software codes are stored in external EPROM (2732A). Static RAM (2Kx8) is used as external transmit/receive buffer. There is no hardware handshaking done between the stations. The serial clock is extracted from the data line using the on-chip phase locked loop.

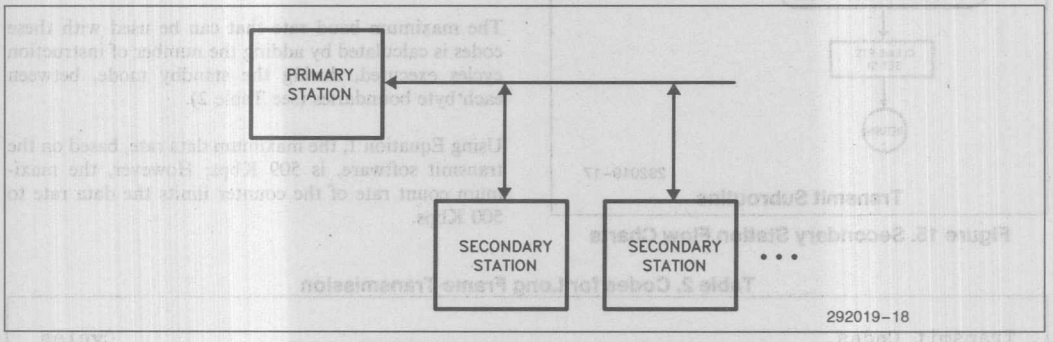


Figure 16. SDLC Multidrop Application Example

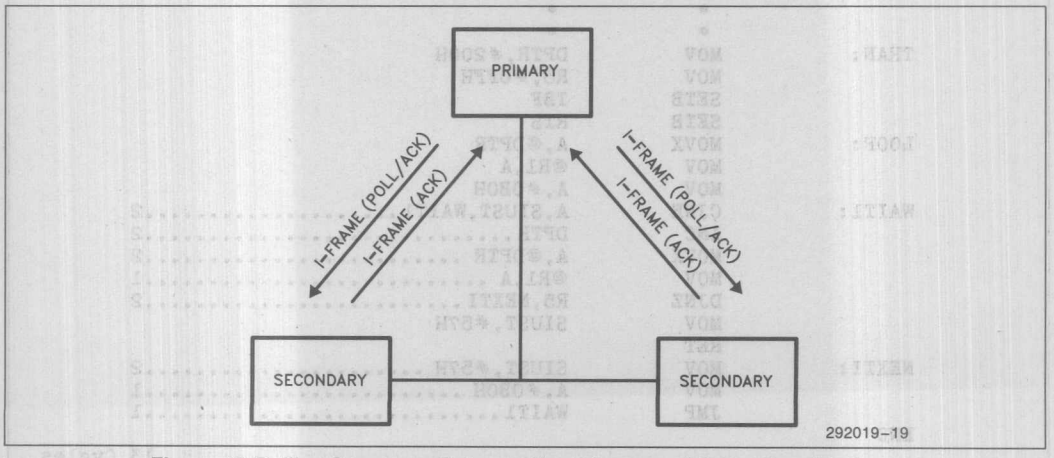


Figure 17. Polling Sequence Between the Primary and Secondary Stations

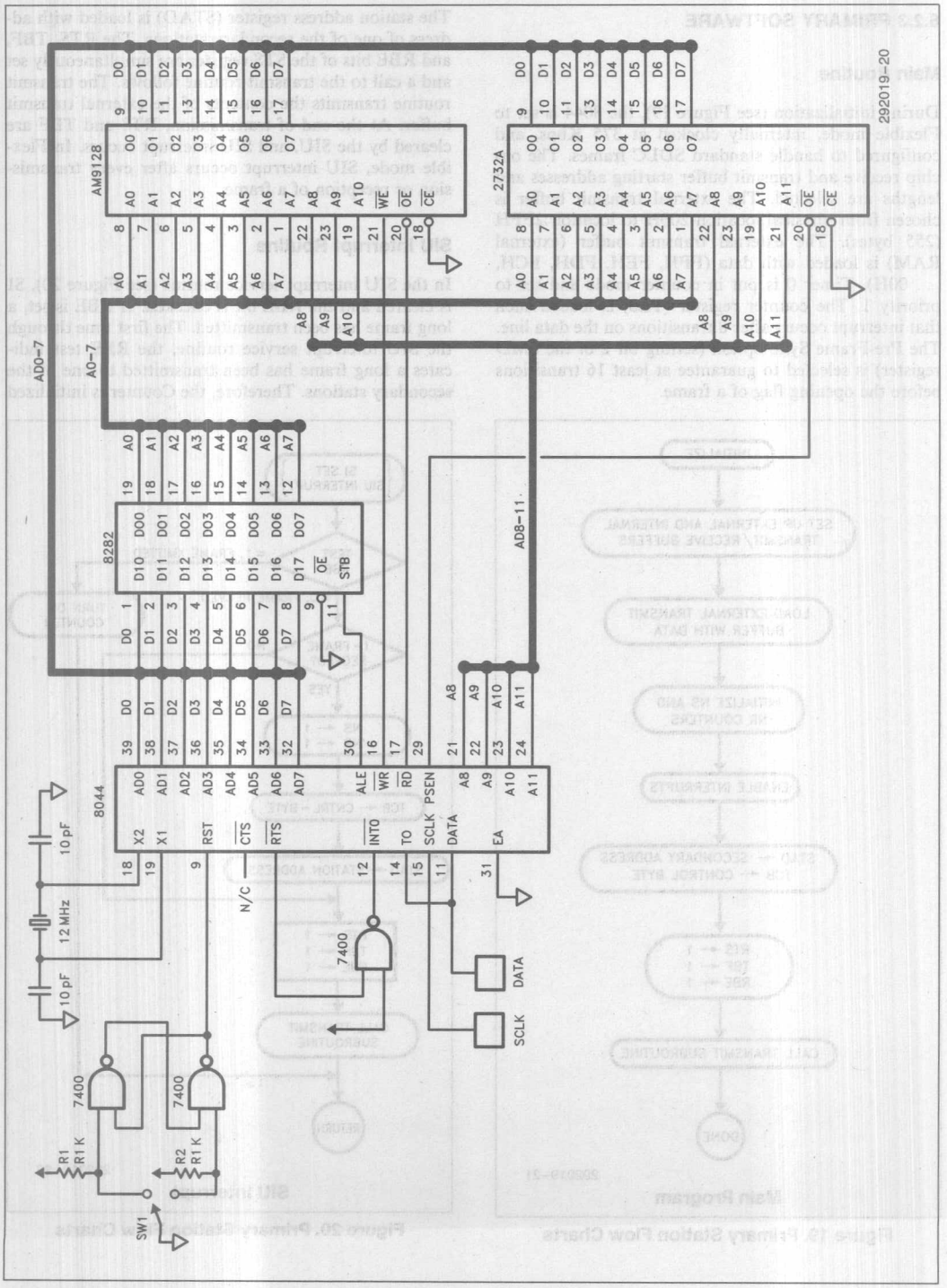


Figure 18. Secondary Station Hardware

6.2.3 PRIMARY SOFTWARE

Main Routine

During initialization (see Figure 19), the 8044 is set to Flexible mode, internally clocked at 375 Kbps, and configured to handle standard SDLC frames. The on-chip receive and transmit buffer starting addresses and lengths are selected. The external transmit buffer is chosen from physical location 200H to location 2FFH (255 bytes). The external transmit buffer (external RAM) is loaded with data (FFH, FEH, FDH, FCH, . . . 00H). Timer 0 is put in counter mode and set to priority 1. The counter register (TL0) is loaded such that interrupt occurs after 8 transitions on the data line. The Pre-Frame Sync option (setting bit 2 of the SMD register) is selected to guarantee at least 16 transitions before the opening flag of a frame.

The station address register (STAD) is loaded with address of one of the secondary stations. The RTS, TBF, and RBE bits of the STS register are simultaneously set and a call to the transmit routine follows. The transmit routine transmits the contents of the external transmit buffer. At the end of transmission, RTS and TBF are cleared by the SIU, and SIU interrupt occurs. In Flexible mode, SIU interrupt occurs after every transmission or reception of a frame.

SIU Interrupt Routine

In the SIU interrupt service routine (see Figure 20), SI is cleared and the RBE bit is checked. If RBE is set, a long frame has been transmitted. The first time through the SIU interrupt service routine, the RBE test indicates a long frame has been transmitted to one of the secondary stations. Therefore, the Counter is initialized

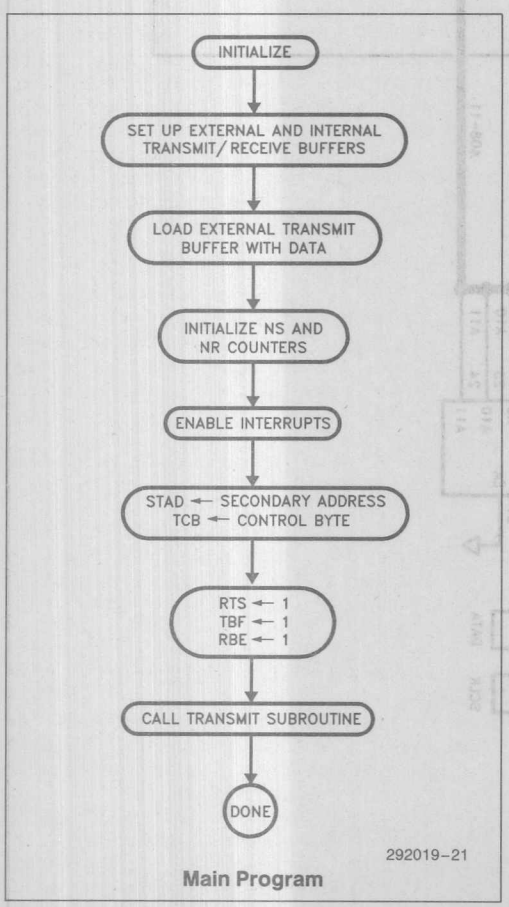


Figure 19. Primary Station Flow Charts

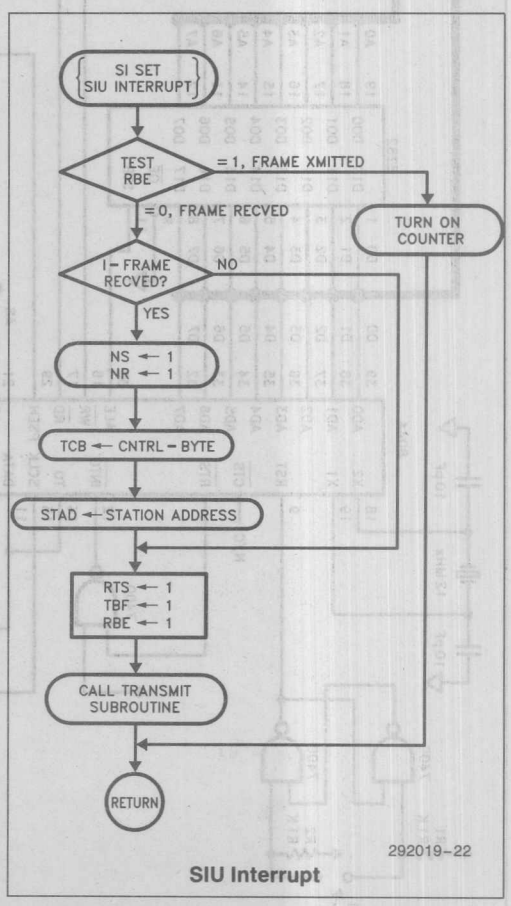


Figure 20. Primary Station Flow Charts

and turned on. The program returns from the interrupt routine before a frame appears on the communication channel.

When a frame appears on the communication line, counter interrupt occurs and the receive routine is executed to move the incoming bytes into the external RAM. After reception of the frame and return from the receive routine, SIU interrupt occurs again.

In the SIU interrupt routine, RBE is checked. Since the RBE bit is clear, a frame has been received. Therefore, the appropriate NS and NR counters are incremented and loaded into the TCB register (two pairs of internal RAM bytes keep track of NS and NR counts for the two secondary stations). Transmission of a frame to the next secondary station is enabled by setting the RTS and the TBF bits. The chip is also put in receive mode (RBE set), and a call to transmit routine is made. After transmission, SIU interrupt occurs again, and the process continues.

6.2.4 SECONDARY SOFTWARE

Main Routine

Both secondary stations have identical software (Appendix B). The only differences are the station addresses. Contents of the STAD register are 55H for one station and 44H for the other.

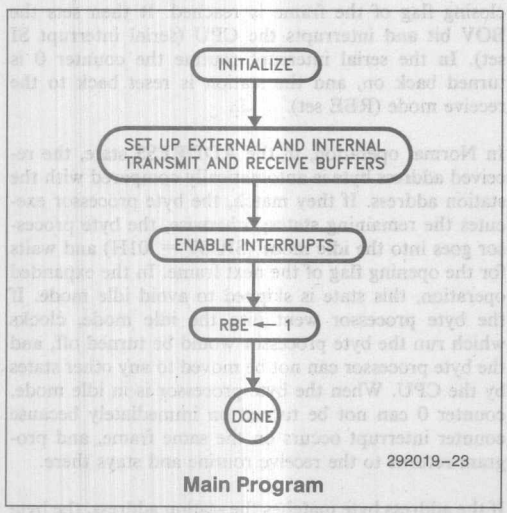


Figure 21. Secondary Station Flow Charts

During initialization, the chip is set to Auto mode, standard SDLC frame, and internally clocked at 375 Kbps (see Figure 21). Internal buffer registers: RBS, RBL, TBS, and TBL are initialized. The RBE bit is set and the counter 0 is turned on.

The secondary is configured to transmit an Information frame every time it is polled. The RTS pin is inverted and tied to INT1 pin. External interrupt 1 is enabled and set to interrupt on low to high transition of the RTS signal. This will cause an interrupt (EX1 set) after a frame is transmitted. In the interrupt routine the CTS pin is cleared to prevent any automatic response from the secondary. If the CTS pin were not disabled, the secondary station would respond with a supervisory frame (RNR) since the TBF is set to zero by the SIU due to the acknowledge. In the SIU interrupt routine, the CTS pin is cleared after the TBF bit is set. If this option is not used, the primary should acknowledge the previously received frame and poll for the next frame in two separate transmissions.

SIU Interrupt Routine

When a frame is received, counter 0 interrupt occurs and the receive routine is executed (see Figure 22). If the incoming frame is addressed to the station, the information bytes are stored in external RAM; Otherwise, the program returns from the receive routine to perform other tasks. At the end of the frame, SIU interrupt occurs. In Auto mode, SIU interrupt occurs whenever an Information frame or a supervisory frame is received. Transmission will not cause an interrupt. In the SIU interrupt service routine, the AM bit of the STS is checked.

If AM bit is set, the interrupt is due to a frame whose address did not match with the address of the station. In this case, NFCS, AM, and the BOV bits are cleared, the RBE bit is set, the counter 0 is initialized and turned on, and program returns from the interrupt routine.

If AM bit is not set, a valid frame has been received and stored in the external RAM. TBF bit is set, CTS pin is activated, counter 0 is disabled and a call to transmit routine is made which transmits the contents of external transmit buffer. This frame also acknowledges the reception of the previously received frame (NS and NR are automatically incremented). Upon return from the transmit routine RBE is set and counter 0 is turned on, thereby putting the chip in the receive mode for another round of data exchange with the primary.

Note that, if the second station is in receive mode, and the counter is enabled and turned on, the CPU will be interrupted each time a frame is on the communication channel. If the frame is not addressed to the secondary station, the chip enters the receive routine, executes only a few lines of code (address comparison) and returns to perform other tasks. This interrupt will not occupy the CPU for more than two data byte periods (43 microseconds at 375 Kbps). At the end of the frame, the BOV bit is set by the SIU, and the SIU interrupt occurs. In the SIU interrupt service routine,

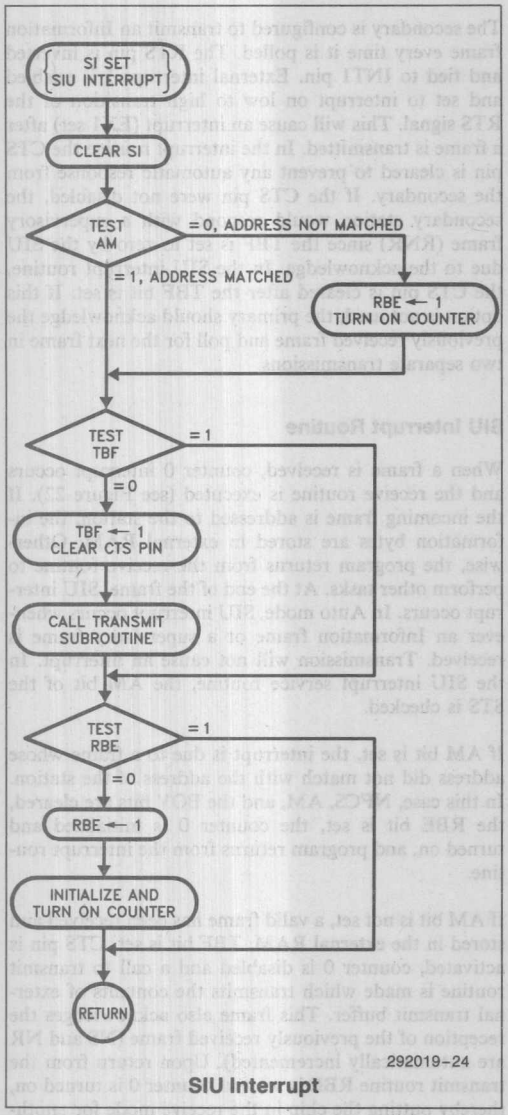


Figure 22. Secondary Station Flow Charts

the RBE bit is set and the counter is turned on which put the chip back in the receive mode.

6.2.5 RECEIVE INTERRUPT ROUTINE

Assembly code for the receive interrupt routine can be found in both primary and secondary software (Appendix B). The receive interrupt routine of the primary station is very similar to that of the primary station in example 1. In the following two sections the receive and transmit routine of the secondary stations are discussed.

In the receive interrupt service routine (see Figure 23), counter 0 is turned off, important registers are saved, receive buffer starting address and receive buffer length of the external RAM are set (do not confuse the external RAM settings with that of the internal RAM buffer.)

After reception of an opening flag, the byte processor jumps to the ADDRESS state and waits until the bit processor processes and moves the receiving address byte to SR. Then, the byte processor is triggered to execute the state. In the secondary stations, the CPU monitors the SIUST register for the ADDRESS state (SIUST = 08H). When the ADDRESS state is reached, the byte processor is moved to the next state (CONTROL state), and the ADDRESS state is skipped. Therefore, when the address byte is moved to SR, the byte processor executes the CONTROL state rather than the ADDRESS state and then jumps to the PUSH-1 state. The execution of the CONTROL state causes the contents of SR (the received address byte) to be loaded into the RCB register.

The CPU checks the contents of RCB with the contents of the STAD (Station Address) register. If they match, the receive routine continues to store the received Information bytes in the external RAM buffer; Otherwise, the byte processor is moved to the very last state (BOV-LOOP), and the program returns from the routine to perform other tasks. The byte processor executes the BOV-LOOP state in each byte boundary until the closing flag of the frame is reached. It then sets the BOV bit and interrupts the CPU (serial interrupt SI set). In the serial interrupt routine the counter 0 is turned back on, and the station is reset back to the receive mode (RBE set).

In Normal operation, in the ADDRESS state, the received address byte is automatically compared with the station address. If they match, the byte processor executes the remaining states; otherwise, the byte processor goes into the idle mode (SIUST = 01H) and waits for the opening flag of the next frame. In the expanded operation, this state is skipped to avoid idle mode. If the byte processor went into the idle mode, clocks which run the byte processor would be turned off, and the byte processor can not be moved to any other states by the CPU. When the byte processor is in idle mode, counter 0 can not be turned on immediately because counter interrupt occurs on the same frame, and program returns to the receive routine and stays there.

If the address byte matches the station address, the byte processor is moved to the CONTROL state again. This time, after execution of the CONTROL state the contents of RCB are the received control byte.

CPU investigates the type of received frame by checking the received control byte. If the receiving frame is not an information frame (i.e. Supervisory frame), execution of receive routine will be terminated to free the

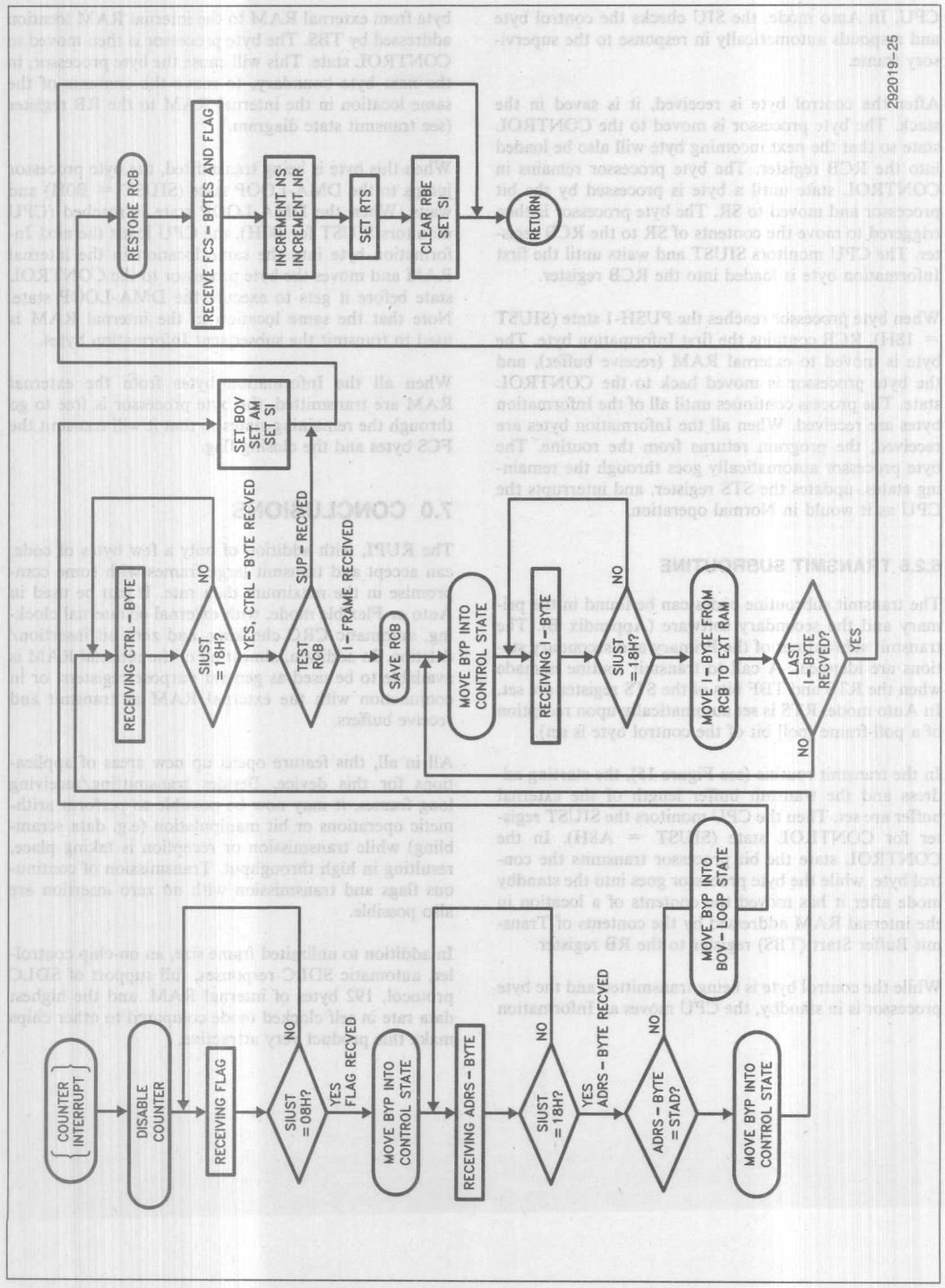


Figure 23. Receive Flow Chart (secondary station)

CPU. In Auto mode, the SIU checks the control byte and responds automatically in response to the supervisory frame.

After the control byte is received, it is saved in the stack. The byte processor is moved to the CONTROL state so that the next incoming byte will also be loaded into the RCB register. The byte processor remains in CONTROL state until a byte is processed by the bit processor and moved to SR. The byte processor is then triggered to move the contents of SR to the RCB register. The CPU monitors SIUST and waits until the first Information byte is loaded into the RCB register.

When byte processor reaches the PUSH-1 state (SIUST = 18H), RCB contains the first Information byte. The byte is moved to external RAM (receive buffer), and the byte processor is moved back to the CONTROL state. The process continues until all of the Information bytes are received. When all the Information bytes are received, the program returns from the routine. The byte processor automatically goes through the remaining states, updates the STS register, and interrupts the CPU as it would in Normal operation.

6.2.6 TRANSMIT SUBROUTINE

The transmit subroutine codes can be found in the primary and the secondary software (Appendix B). The transmit subroutines of the Primary and secondary stations are identical. A call to transmit routine is made when the RTS and TBF bits of the STS register are set. In Auto mode, RTS is set automatically upon reception of a poll-frame (poll bit of the control byte is set).

In the transmit routine (see Figure 15), the starting address and the transmit buffer length of the external buffer are set. Then the CPU monitors the SIUST register for CONTROL state (SIUST = A8H). In the CONTROL state the bit processor transmits the control byte, while the byte processor goes into the standby mode after it has moved the contents of a location in the internal RAM addressed by the contents of Transmit Buffer Start (TBS) register to the RB register.

While the control byte is being transmitted and the byte processor is in standby, the CPU moves an Information byte from external RAM to the internal RAM location addressed by TBS. The byte processor is then moved to CONTROL state. This will cause the byte processor, in the next byte boundary, to move the contents of the same location in the internal RAM to the RB register (see transmit state diagram.)

When this byte is being transmitted, the byte processor jumps to the DMA-LOOP state (SIUST = B0H) and waits. When the DMA-LOOP state is reached (CPU monitors SIUST for B0H), the CPU loads the next Information byte into the same location in the internal RAM and moves the byte processor to the CONTROL state before it gets to execute the DMA-LOOP state. Note that the same location in the internal RAM is used to transmit the subsequent Information bytes.

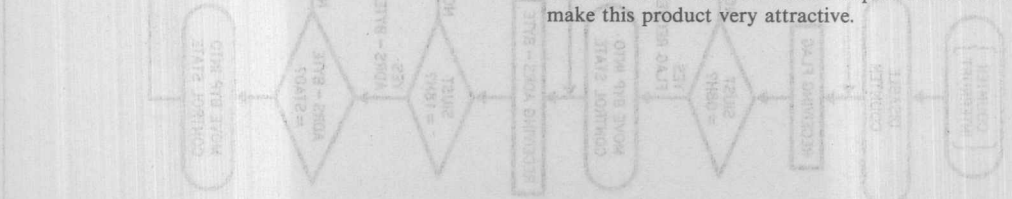
When all the Information bytes from the external RAM are transmitted, the byte processor is free to go through the remaining states so that it will transmit the FCS bytes and the closing flag.

7.0 CONCLUSIONS

The RUPI, with addition of only a few bytes of code, can accept and transmit large frames with some compromise in the maximum data rate. It can be used in Auto or Flexible mode, with external or internal clocking, automatic CRC checking, and zero bit insertion/deletion. In addition, almost all of the internal RAM is available to be used as general purpose registers, or in conjunction with the external RAM as transmit and receive buffers.

All in all, this feature opens up new areas of applications for this device. Besides transmitting/receiving long frames, it may now be possible to perform arithmetic operations or bit manipulation (e.g. data scrambling) while transmission or reception is taking place, resulting in high throughput. Transmission of continuous flags and transmission with no zero insertion are also possible.

In addition to unlimited frame size, an on-chip controller, automatic SDLC responses, full support of SDLC protocol, 192 bytes of internal RAM, and the highest data rate in self clocked mode compared to other chips make this product very attractive.



APPENDIX A
LISTING OF SOFTWARE MODULES FOR APPLICATION EXAMPLE 1

```
$DEBUG NOMOD51
$INCLUDE (REG44.PDF)

;  ASSEMBLY CODE FOR  PRIMARY STATION (POINT TO POINT)
;  FLEXIBLE MODE; FCS OPTION

        ORG   00H               ; LOCATIONS 00 THRU 26H ARE USED
        SJMP  INIT              ; BY INTERRUPT SERVICE ROUTINES.
        ORG   0BH               ; VECTOR ADDRESS FOR TIMER0 INT.
        JMP   REC
        ORG   23H               ; VECTOR ADDRESS FOR SIU INT.
        SJMP  SIINT

;********************* INITIALIZATION **************************

        ORG   26H
INIT:   MOV   SMD,#00000110B    ; EXT CLOCK; PFS=NB=1
        MOV   TBS,#20H          ; INT TRANSMIT BUFFER START
        MOV   TBL,#01H          ; INT TRANSMIT BUFFER LENGTH
        MOV   20H,#55H          ; STATION ADDRESS
        MOV   TMOD,#00000111B   ; COUNTER FUNCTION; MODE 3
        MOV   IE,#10010010B     ; EA=1; SI=1; ET0=1
        MOV   STS,#11100000B    ; TRANSMIT A FRAME
DOT:    SJMP  DOT               ; WAIT FOR AN INTERRUPT

; SIU TRANSMITS THE PFS BYTES, THE OPENNING FLAG, THE CONTENTS
; OF LOCATION 20H, THE CALCULATED FCS-BYTES, AND THE CLOSING
; FLAG.  AT THE END OF TRANSMISSION, SIU INTERRUPT OCCURS.

;************** SERIAL CHANNEL INTERRUPT ROUTINE ****************

SIINT:  CLR    SI
        JNB    RBE,RECVED       ; TRANSMITTED A FRAME ?
        MOV    TL0,#0F8H        ; YES, INITIALIZE COUNTER REGISTER
        MOV    DPTR,#200H       ; EXT RAM RECEIVE BUFFER START
        MOV    R5,#0FFH         ; EXT RAM RECEIVE BUFFER LENGTH
        SETB   TR0              ; TURN ON COUNTER 0
        RETI                    ; RETURN                          292019-28
; WHEN A FRAME APPEARS ON THE SERIAL CHANNEL, COUNTER (RECEIVE)
; INTERRUPT OCCURS. AFTER SERVICING THE INTERRUPT ROUTINE, SIU
; INTERRUPT OCCURS.

RECVED: MOV   STS,#11100000B    ; TRANSMIT A FRAME
        RETI                    ; RETURN


;***************** RECEIVE INTERRUPT ROUTINE ********************

REC:    CLR   TR0               ; DISABLE THE COUNTER 0 INTERRUPT
        MOV   A,#18H            ; PUSH-1 STATE
WAIT1:  CJNE  A,SIUST,WAIT1
NEXTI:  MOV   SIUST,#0EFH       ; MOVE BYP TO CONTROL STATE
        MOV   A,#18H            ; PUSH-1 STATE
WAIT2:  CJNE  A,SIUST,WAIT2     ;
        MOV   A,RCB             ; MOVE RECEIVED BYTE INTO ACC.
        MOVX  @DPTR,A           ; MOVE DATA TO EXT. RAM
        INC   DPTR              ; INCREMENT POINTER TO EXT RAM
        DJNZ  R5,NEXTI          ; LAST BYTE RECEIVED?
        RETI                    ; YES, RETURN
END
                                                                  292019-29
```

```
$DEBUG NOMOD51
$INCLUDE (REG44.PDF)

;  ASSEMBLY CODE FOR SECONDARY STATION (POINT TO POINT)
;  FLEXIBLE MODE; FCS OPTION

       ORG   00H
       SJMP  INIT
       ORG   23H              ; VECTOR ADDRESS FOR SIU INT.
       SJMP  SIINT

;*************** LOAD TRANSMIT BUFFER WITH DATA *************

       ORG   26H
INIT:  MOV   DPTR,#200H       ; EXT RAM XMIT BUFFER START
       MOV   R3,#0FFH         ; EXT RAM XMIT BUFFER LENGHT
LDRAM: MOV   A,R3
       MOVX  @DPTR,A           ; LOAD EXT BUFFER WITH FFH,FEH,...
       INC   DPTR              ; INCREMENT POINTER
       DJNZ  R3,LDRAM

;******************INITIALIZATION ****************************

       MOV   SMD,#00000110B   ; EXT CLOCK; PFS=NB=1
       MOV   R1,#10H
       MOV   TBS,R1           ; INT RAM XMIT BUFFER START
       MOV   TBL,#01H         ; INT RAM XMIT BUFFER LENGTH
       MOV   RBS,#20H         ; INT RAM RECEIVE BUFFER START
       MOV   RBL,#01H         ; INT RAM RECEIVE BUFFER LENGTH
       MOV   STAD,#55H        ; STAD ADDRESS=55H
       MOV   TCON,#00H        ; RESET TCON REGISTER
       MOV   IE,#10010000B    ; ENABLE  SI INT. ;EA=1
       MOV   IP,#0FFH         ; ALL INTERRUPTS: PRIORITY 1
       MOV   STS,#01000000B   ; RBE=1, RECEIVE A FRAME.
DOT:   SJMP  DOT              ; WAIT FOR AN INTERRUPT

; SIU INTERRUPT OCCURS AT THE END OF A RECEIVED FRAME OR
; A TRANSMITTED FRAME.
;*************** SERIAL CHANNEL INTERRUPT ROUTINE ************

SIINT: CLR   SI
       JB    RBE,RETRN        ; RECEIVED A FRAME?
       MOV   A,STAD           ; YES
       CJNE  A,20H,NMACH      ; STATION ADDRESS MATCHED?
       ACALL TRAN             ; YES, CALL TRANSMIT SUBROUTINE

; TRANSMIT SUBROUTINE IS CALLED TO TRANSMIT A LONG FRAME.
; AFTER TRANSMISSION, SI IS SET.  SIU INTERRRUPT IS SERVICED
; AFTER THE CURRENT ROUTINE (SIINT) IS COMPLETED.

NMACH: SETB  RBE              ; RBE=1, RECEIVE A FRAME
RETRN: RETI                   ; RETURN

;************** TRANSMIT SUBROUTINE ***************************

TRAN:  MOV    DPTR,#200H      ; EXT RAM RECEIVE BUFFER START
       MOV   R5,#0FFH         ; EXT RAM RECEIVE BUFFER LENGTH
       SETB  TBF              ; SET TRANSMIT BUFFER FULL
       SETB  RTS              ; ENABLE XMISSION OF AN I-FRAME
LOOP:  MOVX  A,@DPTR          ; MOVE THE 1ST I-BYTE INTO ACC.
       MOV   @R1,A            ; THEN, MOVE TO INT. RAM @ (TBS)
       MOV   A,#0B0H          ; DMA-LOOP STATE
WAIT1: CJNE  A,SIUST,WAIT1    ; WAIT FOR XMISSION OF AN I-FRAME
       INC   DPTR             ; INCREMENT POINTER TO EXT. RAM
       DJNZ  R5,NEXTI         ; ALL BYTES XMITTED?
       MOVX  A,@DPTR          ; YES, EXCEPT THE LAST BYTE.
       MOV   @R1,A            ; MOVE DATA INTO INT. RAM @ (TBS)
       MOV   SIUST,#57H       ; MOVE BYP TO CONTROL STATE
                              ; THE SIU TRANSMITS THE FCS-BYTES
                              ; AND THE CLOSING FLAG.
       RET                    ; RETURN
NEXTI: MOV   SIUST,#57H       ; MOVE BYP TO CONTROL STATE (A8H).
       JMP   LOOP             ; TRANSMIT THE NEXT BYTE
END
```

292019-30

292019-31

APPENDIX B
LISTING OF SOFTWARE MODULES FOR APPLICATION EXAMPLE 2

```
$DEBUG NOMOD51
$INCLUDE (REG44.PDF)

; ASSEMBLY CODE FOR PRIMARY STATION (MULTIPOINT)
; FLEXIBLE MODE; FCS OPTION

        ORG   00H               ; LOCATIONS 00 THRU 26H ARE USED
        SJMP  INIT              ; BY INTERRUPT SERVICE ROUTINES.
        ORG   0BH               ; VECTOR ADDRESS FOR TIMER0 INT.
        JMP   REC
        ORG   23H               ; VECTOR ADDRESS FOR SIU INT.
        SJMP  SIINT

;**************** LOAD TRANSMIT BUFFER WITH DATA ************

       ORG   26H
INIT:  MOV   DPTR,#200H         ; EXT RAM XMIT BUFFER START
       MOV   R3,#0FFH           ; EXT RAM XMIT BUFFER LENGHT
LDRAM: MOV   A,R3
       MOVX  @DPTR,A            ; LOAD BUFFER WITH FFH,FEH,...00
       INC   DPTR               ; INCREMENT POINTER
       DJNZ  R3,LDRAM                                    292019-32
;********************** INITIALIZATION ***********************

        MOV   R0,#0BFH          ; PUT  ZEROS INTO INT. RAM
LOOP:   MOV   A,#00H            ; FROM BFH TO 40H.
        MOV   @R0,A             ; MOVE 0 INTO RAM ADDRESSD BY R0
        DEC   R0
        CJNE  R0,#40H,LOOP
;
        MOV   30H,#00H          ; NS COUNTER FOR STAD=55
        MOV   31H,#00H          ; NR COUNTER FOR STAD=55
        MOV   32H,#0FFH         ; NS COUNTER FOR STAD=44
        MOV   33H,#0FFH         ; NR COUNTER FOR STAD=44
        MOV   34H,#01H          ; PONITER TO SECONDARY STATIONS
        MOV   SMD,#11010100B    ; INT. CLKED @ 375K; NRZI=1; PFS=1
        MOV   RBS,#10H          ; INT. RAM RECEIVE BUFFER START=10H
        MOV   RBL,#00H          ; INT. RAM RECEIVE BUFFER LENGTH=0
        MOV   R1,#20H           ; INT. RAM XMIT BUFFER START=20H
        MOV   TBS,R1
        MOV   TBL,#01H          ; INT. RAM XMIT BUFFER LENGTH=1
        MOV   NSNR,#00H         ; NS=NR=0
        MOV   TMOD,#00000111B   ; COUNTER FUNCTION, MODE 3
        MOV   TCON,#00H
        MOV   IE,#10010010B     ; EA=1; SI=1; ET0=1
        MOV   IP,#00000010B     ; TIMER 0 INT. PRIORITY 1
        MOV   TCB,#00010000B    ; I-FRAME W/POLL
        MOV   STAD,#55H         ; ADDRESS BYTE=55H
        MOV   STS,#11100000B    ; RBE=TBF=RTS=1

; TRANSMIT A LONG FRAME WITH POLL BIT SET, WAIT FOR A
; RESPONSE.

        ACALL TRAN              ; CALL TRANSMIT ROUTINE
DOT:    SJMP  DOT               ; WAIT FOR AN INTERRUPT
                                                          292019-33
```

```
;************** SERIAL INTERRUPT ROUTINE **********************

SIINT:  CLR   SI                ; CLEAR SI
        JB    RBE,RETURN        ; RECEIVED A FRAME ?
        MOV   A,RCB             ; YES, LOAD ACC WITH REC CNTRL BYTE
        JB    ACC.0,GETI        ; IS IT AN I-FRAME ?
        MOV   A,#01H            ; YES
        CJNE  A,34H,SKIP
        MOV   A,30H             ; MOVE NS INTO ACC.
        INC   A                 ; INCREMENT NS
        ANL   A,#00000111B      ; MASK OUT THE LEAST 3 SIG. BITS
        MOV   30H,A             ; SAVE NS
        MOV   A,31H             ; MOVE NR INTO ACC.
        INC   A                 ; INCREMENT NR
        ANL   A,#00000111B      ; MASK OUT THE LEAST 3 SIG. BITS
        MOV   31H,A             ; SAVE NR
        RL    A                 ; SHIFT 4 BITS TO LEFT
        RL    A
        RL    A
        RL    A
        ORL   A,30H             ; MOVE NS COUNT TO ACC.
        RL    A                 ; SHIFT 1 BIT TO LEFT
        ORL   A,#00010000B      ; SET THE POLL BIT
        MOV   TCB,A             ; MOVE CONTROL BYTE INTO TCB REG.
                                ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0
        MOV   STAD,#55H
        MOV   34H,#00H
        JMP   GETI                                          292019-34
SKIP:   MOV   A,32H             ; MOVE NS INTO ACC.
        INC   A                 ; INCREMENT NS
        ANL   A,#00000111B      ; MASK OUT THE LEAST 3 SIG. BITS
        MOV   32H,A             ; SAVE NS
        MOV   A,33H             ; MOVE NR INTO ACC.
        INC   A                 ; INCREMENT NR
        ANL   A,#00000111B      ; MASK OUT THE LEAST 3 SIG. BITS
        MOV   33H,A             ; SAVE NR
        RL    A                 ; SHIFT 4 BITS TO LEFT
        RL    A
        RL    A
        RL    A
        ORL   A,33H             ; MOVE NS COUNT TO ACC.
        RL    A                 ; SHIFT 1 BIT TO LEFT
        ORL   A,#00010000B      ; SET THE POLL BIT
        MOV   TCB,A             ; MOVE CONTROL BYTE INTO TCB
                                ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0
        MOV   STAD,#44H
        MOV   34H,#01H
GETI:   MOV   STS,#11100000B    ; ENABLE TRANSMISSION
        ACALL TRAN              ; CALL TRANSMIT ROUTINE
        RETI
RETURN: CLR   EA                ; DISABLE ALL INTERRUPTS
        MOV   TL0,#0FBH         ; INTERRUPT AFTER 8 COUNTS
        SETB  TR0               ; TURN ON COUNTER 0
        SETB  EA
        RETI                                                292019-35
;************** RECEIVE INTERRUPT ROUTINE ********************

REC:    CLR   TR0               ; TURN OFF COUNTER 0
        MOV   DPTR,#400H        ; EXT. RAM RECEIVE BUFFER START
        MOV   R5,#0FFH          ; EXT. RAM RECEIVE BUFFER LENGTH
        MOV   A,#18H            ; PUSH-1 STATE
WAIT1:  CJNE  A,SIUST,WAIT1     ; WAIT FOR THE CONTROL BYTE
        PUSH  RCB               ; SAVE RECEIVE CONTROL BYTE
NEXTI:  MOV   SIUST,#0EFH       ; PUSH "BYP" INTO CONTROL STATE(10H).
        MOV   A,#18H            ; PUSH-1 STATE
WAIT2:  CJNE  A,SIUST,WAIT2     ; WAIT FOR AN I-BYTE
        MOV   A,RCB             ; MOVE RECEIVED I-BYTE INTO ACC.
        MOVX  @DPTR,A           ; MOVE DATA TO EXT. RAM
        INC   DPTR              ; INCREMENT PTR TO EXTERNAL RAM
        DJNZ  R5,NEXTI          ; IS IT THE LAST I-BYTE?
        POP   RCB               ; YES, RESTORE THE CONTENTS OF RCB
        RETI                    ; RETURN

;***************** TRANSMIT SUBROUTINE **********************

TRAN:   MOV   DPTR,#200H        ; EXT. RAM TRANSMIT BUFFER START
        MOV   R5,#0FFH          ; EXT. RAM TRANSMIT BUFFER LENGTH
        MOV   A,#0A8H           ; CONTROL STATE
WAIT:   CJNE  A,SIUST,WAIT      ; WAIT  FOR CTRL BYTE XMISSION
        MOVX  A,@DPTR           ; MOVE DATA FROM EXT. RAM TO ACC.
        MOV   @R1,A             ; MOVE DATA INTO INT. RAM @ (TBS)
        INC   DPTR              ; INCREMENT POINTER
        DJNZ  R5,NXTI           ; IS IT THE LAST I-BYTE ?
        MOV   SIUST,#57H        ; NO. XMIT THE LAST I-BYTE
        RET                     ; RETURN.
NXTI:   MOV   SIUST,#57H        ; KEEP "BYP" IN CONTROL STATE(A8H).
        MOV   A,#0B0H           ; DMA-LOOP STATE
        JMP   WAIT              ; TRANSMIT THE NEXT BYTE
END
                                                            292019-36
```

```
$DEBUG NOMOD51
$INCLUDE (REG44.PDF)

; ASSEMBLY CODE FOR SECONDARY STATIONS (MULTIPOINT)
; AUTO MODE; FCS OPTION

        ORG   00H
        SJMP  INIT
        ORG   0BH               ; VECTOR ADDRESS FOR TIMER0 INT.
        JMP   REC
        ORG   13H               ; VECTOR ADDRESS FOR EXT. INT. 1
        JMP   XINT1
        ORG   23H               ; VECTOR ADDRESS FOR SIU INTERRUPT
        JMP   SIINT

;********************INITIALIZATION ***************************

        ORG   26H
INIT:   MOV   SMD,#11010100B    ; INT. CLKED @ 375K;NRZI=1;PFS=1
        MOV   STAD,#55H         ; STATION ADDRESS; STAD=44H FOR THE
                                ; OTHER STATION
        MOV   RBS,#10H          ; INT. RAM RECEIVE BUFFER START
        MOV   RBL,#00H          ; INT. RAM RECEIVE BUFFER LENGTH
        MOV   R1,#20H
        MOV   TBS,R1            ; INT. RAM XMIT BUFFER START
        MOV   TBL,#01H          ; INT. RAM XMIT BUFFER LENGTH
        MOV   NSNR,#00H         ; NS=NR=0
        MOV   TCON,#00000100B   ; EXT. INT.: EDGE TRIGGERED
        MOV   IE,#00010110B     ; SI=1; ET0=1; EX0=1
        MOV   IP,#00000010B     ; TIMER 0: PRIORITY 1
        MOV   TMOD,#0000011B    ; COUNTER FUNCTION: MODE 3
        MOV   STS,#01000010B    ; RECEIVE I-FRAME.
        MOV   TL0,#0F8H         ; SET COUNTER TO OVERFLOW
                                ; AFTER 8 COUNTS
        SETB  TR0               ; TURN ON COUNTER
        SETB  EA                ; ENABLE ALL INTERRUPTS
DOT:    SJMP  DOT               ; WAIT FOR AN INTERRUPT.
; CPU IS INTERRUPTED  AT THE END OF RECEPTION (SI SET), AND AT*
; THE END OF LONG-FRAME TRANSMISSION (EX0 SET).               *

;***************EXTERNAL INTERRUPT ***************************

XINT1:  SETB  P1.7              ; DISABLE CTS PIN
        RETI                    ; RETURN.

;*************** SERIAL INTERRUPT ROUTINE *********************

SIINT:  CLR   SI
        JB    AM,HOP            ; ADDRESS MATCHED?
        CLR   EA                ; DISABLE ALL INTERRUPTS
        MOV   STS,#01000010B    ; RBE=1; NB=1
        MOV   TL0,#0F8H
        SETB  TR0               ; TURN ON COUNTER 0
        SETB  EA                ; ENABLE ALL INTERRUPTS
        RETI                    ; RETURN.
;
HOP:    JB    TBF,GETI          ; A FRAME TRANSMITTED?
        SETB  TBF               ; ENABLE TRANSMISSION OF I-FRAME
        CLR   P1.7              ; ENABLE CTS PIN
        ACALL TRAN              ; CALL TRANSMIT ROUTINE
GETI:   JB    RBE,RETURN        ; A FRAME RECEIVED?
        CLR   EA                ; DISABLE ALL INTERRUPTS
        SETB  RBE               ; PUT RUPI IN RECEIVE MODE
        MOV   TL0,#0F8H         ;
        SETB  TR0               ; TURN ON COUNTER 0
        SETB  EA                ; ENABLE ALL INTERRUPTS
RETURN: RETI                    ; RETURN.
;***************** TRANSMIT SUBROUTINE ************************

TRAN:   MOV   DPTR,#200H        ; EXT. RAM TRANSMIT BUFFER START
        MOV   R5,#0FFH          ; EXT. RAM TRANSMIT BUFFER LENGTH
        MOV   A,#0A8H           ; CONTROL STATE
WAIT:   CJNE  A,SIUST,WAIT      ; WAIT  FOR CONTROL BYTE TRANSMISSION
        MOVX  A,@DPTR           ; MOVE DATA FROM EXT. RAM TO ACC.
        MOV   @R1,A             ; MOVE DATA INTO INT. RAM AT @TBS
        INC   DPTR              ; INCREMENT POINTER
        DJNZ  R5,NXTI           ; IS IT THE LAST I-BYTE ?
        MOV   SIUST,#57H        ; XMIT THE LAST I-BYTE
        RET                     ; RETURN.
NXTI:   MOV   SIUST,#57H        ; KEEP "BYP" IN CONTROL STATE
        MOV   A,#0B0H           ; DMA-LOOP STATE
        JMP   WAIT              ; TRANSMIT THE NEXT BYTE
```

292019-37

292019-38

292019-39

```
;**********RECEIVE INTERRUPT ROUTINE**************************

REC:    CLR   TR0              ; TURN OFF COUNTER 0
        MOV   DPTR,#200H       ; EXT. RAM RECEIVE BUFFER START
        MOV   R5,#0FFH         ; EXT. RAM RECEIVE BUFFER LENGTH
        MOV   A,#08H           ; ADDRESS STATE
HOLD:   CJNE  A,SIUST,HOLD     ; WAIT FOR ADDRESS BYTE
        MOV   SIUST,#0EFH      ; MOVE "BYP" INTO CONTROL STATE
                               ; SKIP THE ADDRESS STATE
        MOV   A,#18H           ; PUSH-1 STATE
WAIT1:  CJNE  A,SIUST,WAIT1    ; WAIT FOR THE ADDRESS BYTE
        MOV   A,RCB            ; MOVE THE RECEIVED ADDRESS BYTE TO ACC.
        CJNE  A,STAD,WAIT2     ; ADDRESS MATCHED?
        SJMP  WAIT3            ; YES.
WAIT2:  MOV   RCB,#00010000B   ; MOVE INFO. CONTROL BYTE TO RCB
        MOV   SIUST,#0CFH      ; MOVE "BYP" INTO BOV-LOOP STATE
        RETI                   ; RETURN
;
WAIT3:  MOV   SIUST,#0EFH      ; MOVE "BYP" INTO CONTROL STATE
        MOV   A,#18H           ; PUSH-1 STATE
WAIT4:  CJNE  A,SIUST,WAIT4    ; WAIT FOR THE CONTROL BYTE
        MOV   A,RCB            ; MOVE RECEIVE CONTROL BYTE INTO ACC.
        JB    ACC.0,RTRN       ; IF NOT AN I-FRAME RETURN
        PUSH  RCB              ; SAVE RECEIVE CONTROL BYTE
NEXTI:  MOV   SIUST,#0EFH      ; PUSH "BYP" INTO CONTROL STATE(10H).
        MOV   A,#18H           ; PUSH-1 STATE
WAIT5:  CJNE  A,SIUST,WAIT5    ; WAIT FOR AN I-BYTE
        MOV   A,RCB            ; MOVE RECEIVED I-BYTE INTO ACC.
        MOVX  @DPTR,A          ; MOVE DATA TO EXT. RAM
        INC   DPTR             ; INCREMENT PTR TO EXTERNAL RAM
        DJNZ  R5,NEXTI         ; IS IT THE LAST I-BYTE?
        POP   RCB              ; YES. RESTORE THE CONTENTS OF RCB
RTRN:   RETI                   ; RETURN
END
```

292019-40

intel

ARTICLE REPRINT

AR-307

NOVEMBER 1983

MICROCONTROLLER WITH INTEGRATED HIGH PERFORMANCE COMMUNICATIONS INTERFACE

CHARLES YAGER
APPLICATIONS ENGINEER

© INTEL CORPORATION

January 1985
ORDER NUMBER 230876-001
230876-1

SUMMARY

The 8044 offers a lower cost and higher performance solution to networking microcontrollers than conventional solutions. The system cost is lowered by integrating an entire microcomputer with an intelligent HDLC/SDLC communication processor onto a single chip. The higher performance is realized by integrating two processors running concurrently on one chip; the powerful 8051 microcontroller and the Serial Interface Unit. The 8051 microcontroller is substantially off-loaded from the communication tasks when using the AUTO mode. In the AUTO mode the SIU handles many of the data link functions in hardware. The advantages of the AUTO mode are: less software is required to implement a secondary station data link, the 8051 CPU is offloaded, and the turn-around time is reduced, thus increasing the network throughput. Currently the 8044 is the only microcontroller with a sophisticated communications processor on-chip. In the future there will be more microcontrollers available following this trend.

INTRODUCTION

Today microcontrollers are being designed into virtually every type of equipment. For the household, they are turning up in refrigerators, thermostats, burglar alarms, sprinklers, and even water softeners. At work they are found in laboratory instruments, copiers, elevators, hospital equipment, and telephones. In addition, a lot of microcomputer equipment contains more than one microcontroller. Applications using multiple microcontrollers as well, as the office and home, are now faced with the same requirements that laboratory instruments were faced with 12 years ago — they need to connect them together and have them communicate. This need was satisfied in the laboratory with the IEEE-488 General Purpose Instrumentation Bus (GPIB). However, GPIB does not meet the current design objectives for networking microcontrollers.

Today there are many communications schemes and protocols available; some of the popular ones are GPIB, Async, HDLC/SDLC, and Ethernet. Common design objectives of today's networks are: low cost, reliable, efficient throughput, and expandable. In examining available solutions, GPIB does not meet these design objectives; first, the cable is too expensive (parallel communications), second, it can only be used over a limited distance (20 meters), and third, it can only handle a limited number of stations. For general networking, serial communications is preferable because of lower cable costs and higher reliability (fewer connections). While Ethernet provides very high performance, it is more of a system backbone rather than a microcontroller interconnect. Async, on the other hand, is inexpensive but it is not an efficient protocol for data block or file transfers. Even with some new modifications such as a 9 bit protocol for addressing, important functions such as acknowledgements, error checking/recovery, and data transparency are not standardized nor supported by available data comm chips.

SDLC, Synchronous Data Link Control, meets the requirements for communications link design. The physical medium can be used on two or four wire twisted pair with inexpensive transceivers and connectors. It can also be interfaced through modems, which allows it to be used on broadband networks, leased or switched telephone lines. VLSI controllers have been available from a number of vendors for years; higher performance and more user friendly SDLC controllers continue to appear. SDLC has also been designed to be very reliable. A 16 bit CRC checks the integrity of the received data, while frame numbering and acknowledgements are also built in. Using SDLC, up to 254 stations can be uniquely addressed, while HDLC addressing is unlimited. If an RS-422 only requires a single +5 volt power supply.

What will the end user pay for the added value provided by communications? The cost of the communications hardware is not the only additional cost. There will be performance degradation in the main application because the microcontroller now has additional tasks to perform. There are two extremes to the cost of adding communication capability. One could spend very little by adding an I/O port and have the CPU handle everything from the baud rate to the protocol. Of course the main application would be idle while the CPU was communicating. The other extreme would be to add another microcontroller to the system dedicated to communications. This communications processor could interface to the main CPU through a high speed parallel link or dual port RAM. This approach would maintain system performance, but it would be costly.

Adding HDLC/SDLC Networking Capability

Figure 1 shows a microcomputer system with a conventional HDLC/SDLC communications solution. The additional hardware needed to realize the conventional design is: an HDLC/SDLC communication chip, additional ROM for the communication software, part of an interrupt controller, a baud rate generator, a phase locked loop, NRZI encoded decoder, and a cable driver locked loop are used when the transmitter does not send the clock on a separate line from the data (i.e. over telephone lines, or two wire cable), the NRZI encoder decoder is used in HDLC/SDLC to combine the clock into the data line. A phase locked loop is used to recover the clock from the data line.

The majority of the available communication chips provide a limited number of data link control functions. Most of them will handle Zero Bit Insertion/Deletion (ZBI/D), Flags, Aborts, Automatic

230876-2

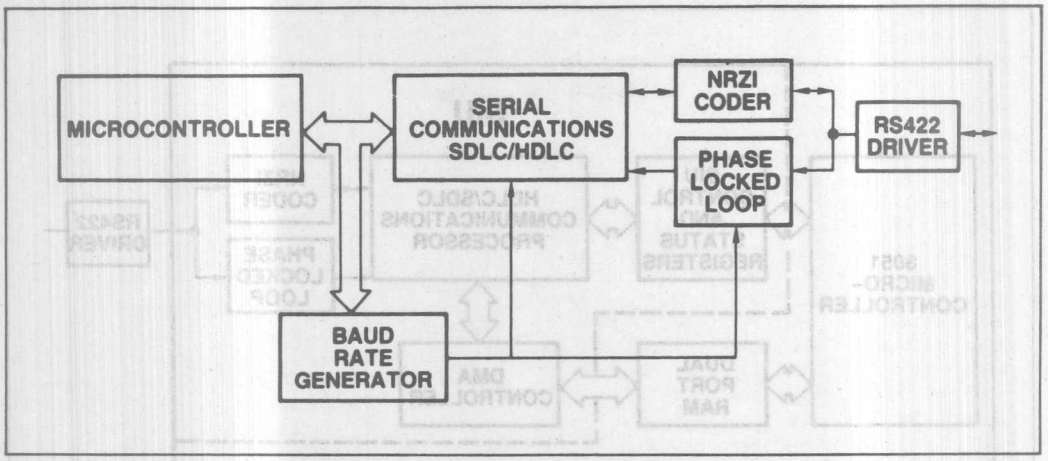


Figure 1. Conventional microcontroller networking solution

address recognition, and CRC generation and checking. It is the CPU's responsibility to manage link access, command recognition and response, acknowledgements and error recovery. Handling these tasks can take a lot of CPU time. In addition, servicing the transmission and reception of data bytes can also be very time consuming depending on the method used.

Using a DMA controller can increase the overall system performance, since it can transfer a block of data in fewer clock cycles than a CPU. In addition, the CPU and the DMA controller can multiplex their access to the bus so that both can be running at virtually the same time. However, both the DMA controller and the CPU are sharing the same bus, therefore, neither one get to utlize 100% of the bus bandwidth. Microcontrollers available today do not support DMA, therefore, they would have to use interrupts, since polling is unacceptable in a multitasking environment.

In an interrupt driven, the CPU has overhead in addition to servicing the interrupt. During each interrupt request the CPU has to save all of the important registers, transfer a byte, update pointers and counters, then restore all of its registers. At low bit rates this overhead may be insignificant. However, the percentage of overhead increases linearly with the bit rate. At high bit rates this overhead would consume all of the CPU's time. There is another nuisance factor associated with interrupt driven systems, interrupt latency. Too much interrupt latency will cause data to be lost from underrun and overrun errors.

The additional hardware necessary to implement the communications solution, as shown in Figure 1, would require 1 LSI chip and about 10 TTL chips. The cost of CPU throughput degradation can be even greater. The percentage of time the CPU has to spend servicing the communication tasks can be anywhere from 10-100%, depending on the serial bit rate. These high costs will prevent consumer acceptance of networking microcomputer equipment.

A Highly Integrated, High Performance Solution

The 8044 reduces the cost of networking microcontrollers without compromising performance. It contains all of the hardware components necessary to implement a microcomputer system with communications capability, plus it reduces the CPU and software overhead of implementing HDLC/SDLC. Figure 2 shows a functional block diagram of the 8044.

The 8044 integrates the powerful 8051 microcontroller with an intelligent Serial Interface Unit to provide a single chip solution which efficiently implements a distributed processing or distributed control system. The microcontroller is a self sufficient unit containing ROM, RAM, ALU and its own peripherals. The 8044's architecture and instruction set are identical to the 8051's. The Serial Interface Unit (SIU) uses bit synchronous HDLC/SDLC protocol and can communicate at bit rates up to 2.4 Mbps, externally clocked, or up to 375 Kbps using the on-chip digital phase locked loop. The SIU contains its own processor, which operates concurrently with the microcontroller.

The CPU and the SIU, in the 8044, interface through 192 bytes of dual port RAM. There is no hardware arbitration in the dual port RAM. Both processor's memory access cycles are interlaced; each processor has access every other clock cycle. Therefore, there

230876-3

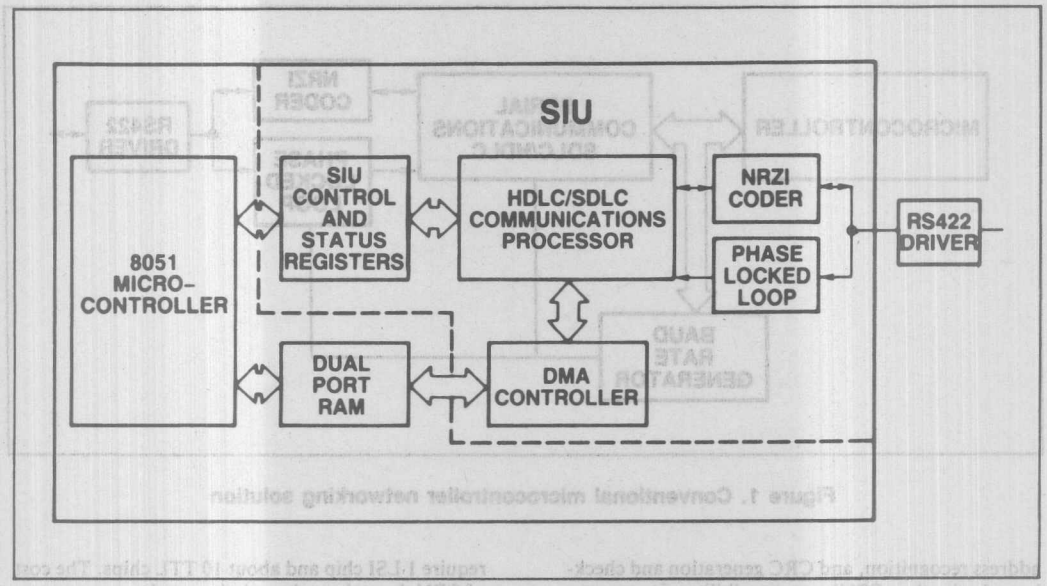


Figure 2. 8044 single chip microcontroller networking solution

is no throughput loss in either processor as a result of the dual port RAM, and execution times are deterministic. Since this has always been the method for memory access on the 8051 microcontroller, 8051 programs have the same execution time in the 8044.

By integrating all of the communication hardware onto the 8051 microcontroller, the hardware cost of the system is reduced. Now several chips have been integrated into a single chip. This means that the system power is reduced, P.C. board space is reduced, inventory and assembly is reduced, and reliability is improved. The improvement in reliability is a result of fewer chips and interconnections on the P.C. board.

As mentioned before, there can be two extremes in a design which adds communications to the microcomputer system. The 8044 solution uses the high end extreme. The SIU on the 8044 contains its own processor which communicates with the 8051 processor through dual port RAM and control/status registers. While the SIU is not a totally independent communications processor, it substantially offloads the 8051 processor from the communication tasks.

The DMA on the 8044 is dedicated to the SIU. It cannot access external RAM. By having a DMA controller in the SIU, the 8051 CPU is offloaded. As a result of the dual port RAM design, the DMA does not share the running at full speed while the frames are being transmitted or received. Also, the nuisance of overrun and underrun errors is totally eliminated since the dedicated DMA controller is guaranteed to meet the maximum data rates. Having a dedicated DMA controller means that the serial channel interrupt can be the lowest priority, thus allowing the CPU to have higher priority real time interrupts.

Figure 3 shows a comparison between the conventional and the 8044 solution on the percentage of time the CPU must spend sending data. This diagram was derived by assuming a 64 byte information frame is being transmitted repeatedly. The conventional solution is interrupt driven, and each interrupt service routine is assumed to take about 15 instructions with a 1 μsec instruction cycle time. At 533 Kbps, an interrupt would occur every 15 usec. Thus, the CPU becomes completely dedicated to servicing the serial communications. The conventional design could not support bit rates higher than this because of underruns and overruns. For the 8044 to repeatedly send 64 byte frames, it simply has to reinitialize the DMA controller. As a result, the 8044 can support bit rates up to 2.4 Mbps.

Some of the other communications tasks the CPU has to perform, such as link access, command recognition/response, and acknowledgements, are performed automatically by the SIU in a mode called "AUTO." The combination of the dedicated DMA controller and the AUTO mode, substantially offload

230876-4

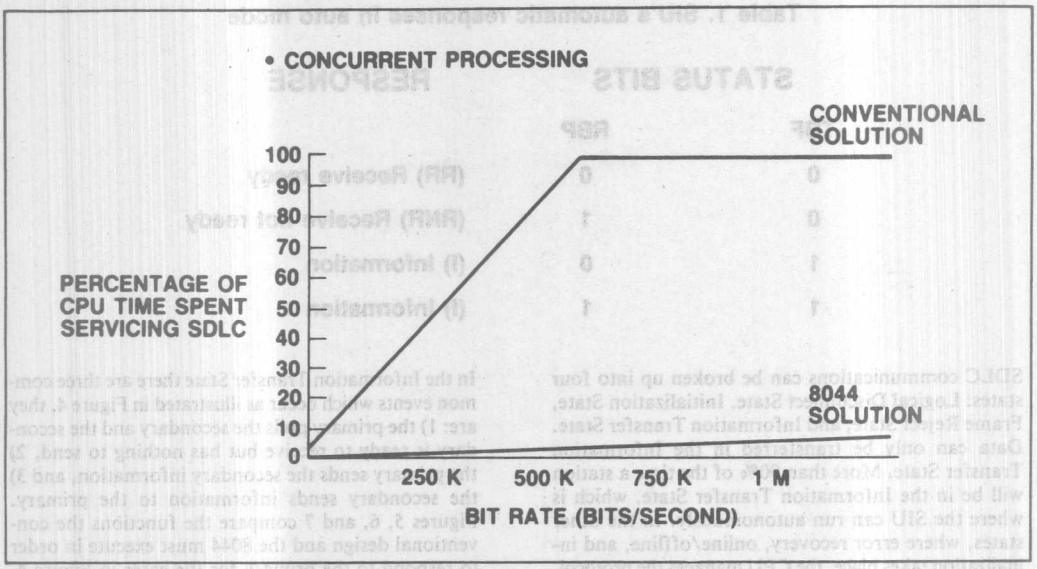


Figure 3. SIU offloads CPU

the CPU, thus allowing it to devote more of its power to other tasks.

8044's Auto Mode

In the AUTO mode the SIU implements in hardware a subset of the SDLC protocol such that it responds to many SDLC commands without CPU intervention. All AUTO mode responses to the primary station conform to IBM's SDLC definition. In the AUTO mode the 8044 can only be a secondary station operating in SDLC specified "Normal Response Mode." Normal Response Mode means that the secondary station can not transmit unless it is polled by the primary station. The SIU in the AUTO mode can recognize and respond to the following SDLC commands without CPU intervention: I (Information), RR (Receive Ready), RNR (Receive Not Ready), REJ (Reject), and for loop mode UP (Unnumbered Poll). The SIU can generate the following responses without CPU intervention: I, RR, and RNR. In addition, the SIU manages Ns and Nr in the control field. If it detects an error in either Ns or Nr, it interrupts the CPU for error recovery.

How does the SIU know what responses to send to the primary? It uses two status bits which are set by the CPU. The two bits are TBF (Transmit Buffer Full) and RBP (Receive Buffer Protect). TBF indicates that the CPU wants to send data, and RBP indicates that the receive data buffer is full. Table 1 shows the responses the SIU will send based on these two status bits. This is an innovative approach to communication design. The CPU in the 8044 with one instruction can directly set a bit which communicates to the primary what its transmit and receive buffering status is.

When the CPU wants to send a frame, it loads the transmit buffer with the data, loads the starting address and the count of the data into the SIU, then sets TBF to transmit the frame. The SIU waits for the primary station to poll it with a RR command. After the SIU is polled, it automatically sends the information frame to the primary with the proper control field. The SIU then waits for a positive acknowledgement from the primary before incrementing the Ns field and interrupting the CPU for more data. If a negative acknowledgement is received, the SIU automatically retransmits the frame.

When the 8044 is ready to receive information, the CPU loads the receive buffer starting address and the buffer length into the SIU, then enables the receiver. When a valid information frame with the correct address and CRC is received, the SIU will increment the Nr field, disable the receiver and interrupt the CPU indicating that a good I frame has been received. The CPU then sets RBP, reenables the receiver and processes the received data. By enabling the receiver with RBP set, the SIU will automatically respond to polls with a Receive Not Ready, thus keeping the link moving rather than timing out the primary from a disabled receiver, or interrupting the CPU with another poll before it has processed the data. After the data has been processed, the CPU clears RBP, returning to the Receive Ready responses.

230876-5

Table 1. SIU's automatic responses in auto mode

STATUS BITS		RESPONSE
TBF	RBP	
0	0	(RR) Receive ready
0	1	(RNR) Receive not ready
1	0	(I) Information
1	1	(I) Information

SDLC communications can be broken up into four states: Logical Disconnect State, Initialization State, Frame Reject State, and Information Transfer State. Data can only be transferred in the Information Transfer State. More than 90% of the time a station will be in the Information Transfer State, which is where the SIU can run autonomously. In the other states, where error recovery, online/offline, and initialization takes place, the CPU manages the protocol.

In the Information Transfer State there are three common events which occur as illustrated in Figure 4, they are: 1) the primary polls the secondary and the secondary is ready to receive but has nothing to send, 2) the primary sends the secondary information, and 3) the secondary sends information to the primary. Figures 5, 6, and 7 compare the functions the conventional design and the 8044 must execute in order to respond to the primary for the cases in Figure 4.

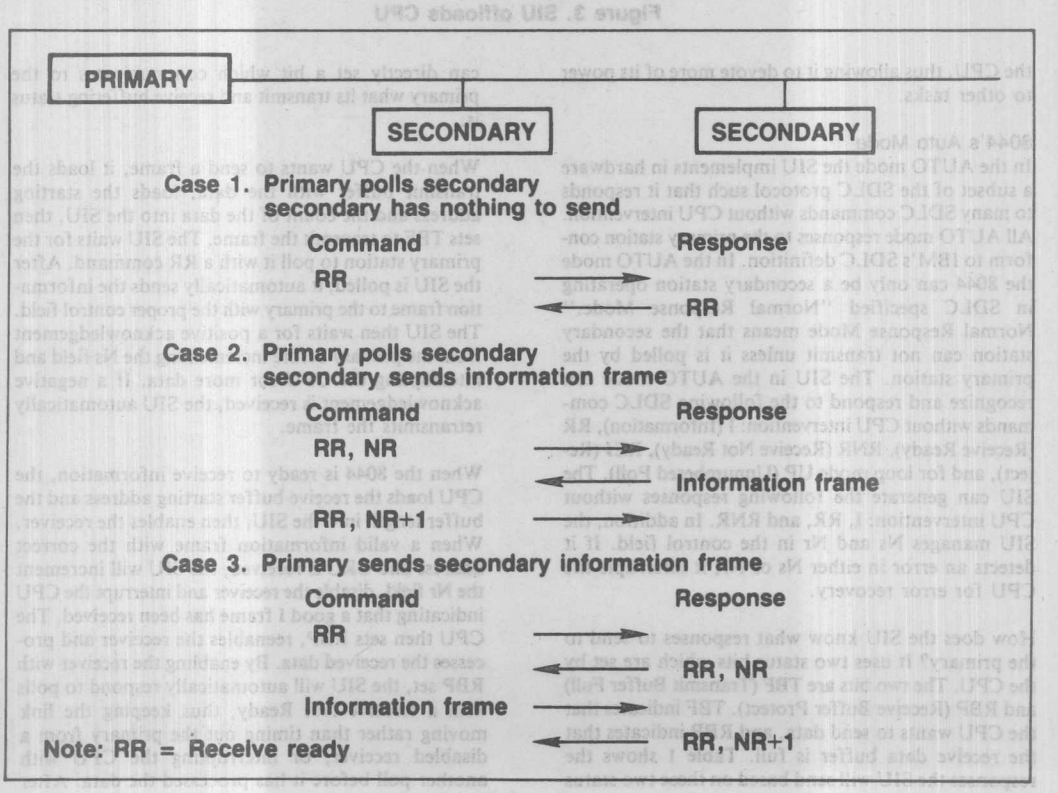


Figure 4. SDLC commands and responses in the information transfer state

230876-6

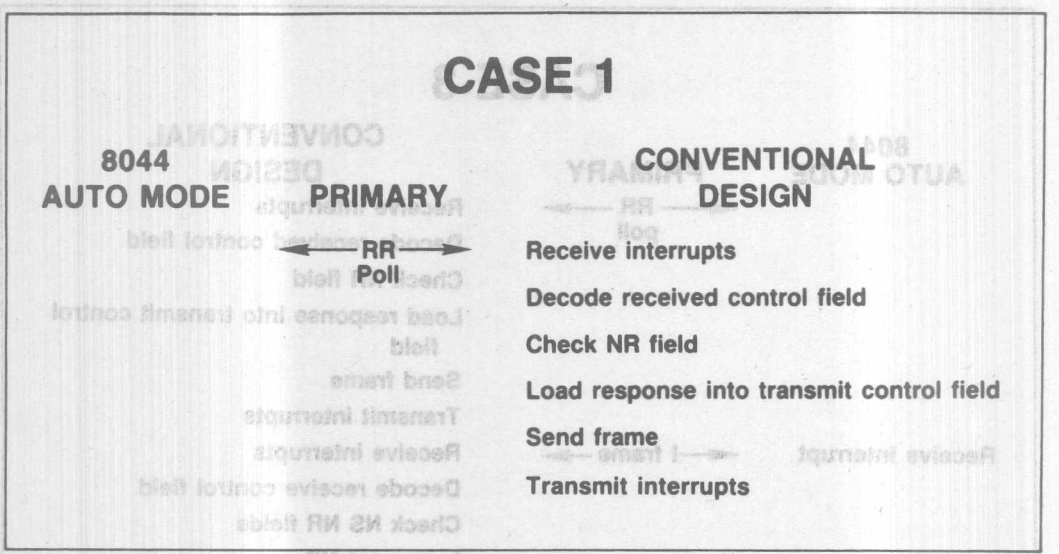


Figure 5. Primary polls secondary, secondary has nothing to send

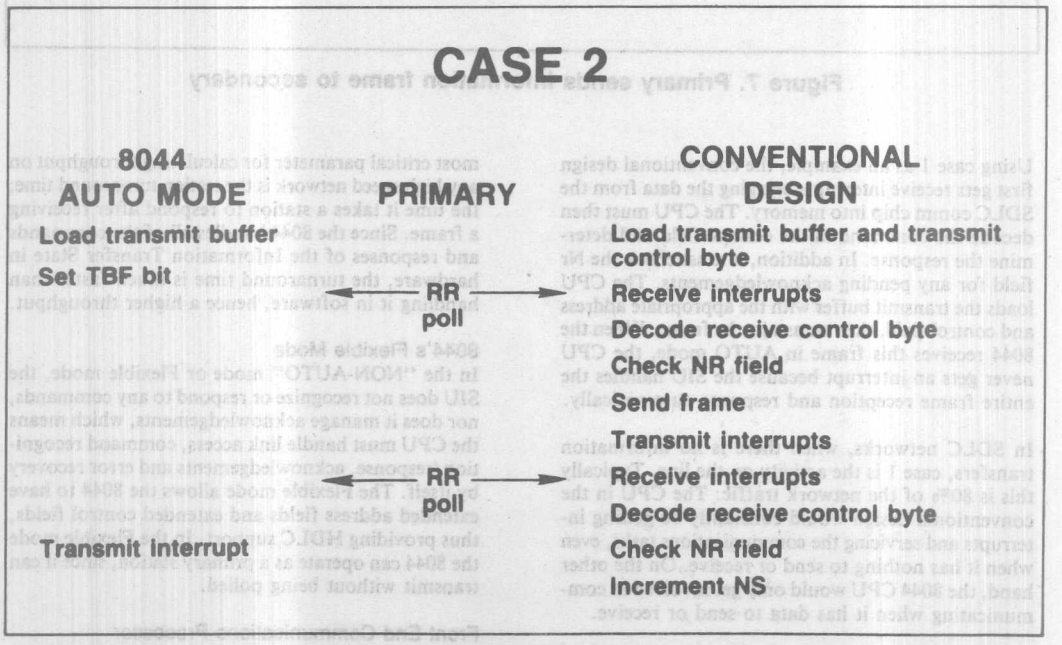


Figure 6. Primary polls secondary, secondary sends information frame

230876-7

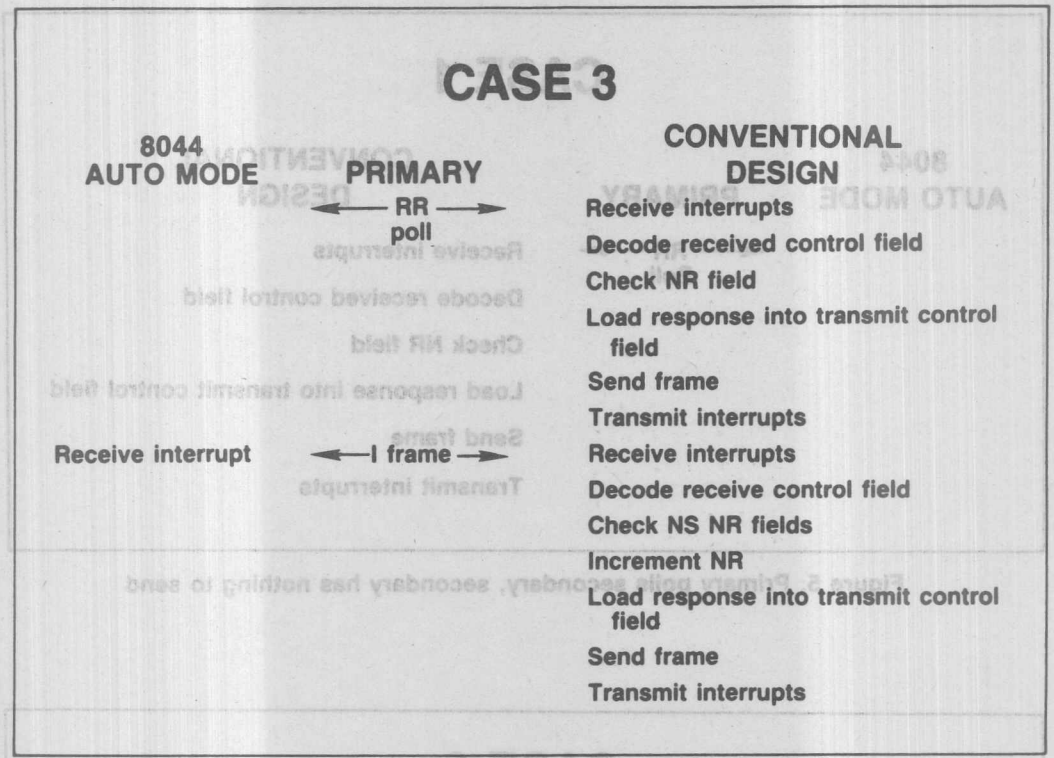


Figure 7. Primary sends information frame to secondary

Using case 1 as an example, the conventional design first gets receive interrupts bringing the data from the SDLC comm chip into memory. The CPU must then decode the command in the control field and determine the response. In addition, it must check the Nr field for any pending acknowledgements. The CPU loads the transmit buffer with the appropriate address and control field, then transmits the frame. When the 8044 receives this frame in AUTO mode, the CPU never gets an interrupt because the SIU handles the entire frame reception and response automatically.

In SDLC networks, when there is no information transfers, case 1 is the activity on the line. Typically this is 80% of the network traffic. The CPU in the conventional design would constantly be getting interrupts and servicing the communications tasks, even when it has nothing to send or receive. On the other hand, the 8044 CPU would only get involved in communicating when it has data to send or receive.

Having the SIU implement a subset of the SDLC protocol in hardware not only offloads the CPU, but it also improves the throughput on the network. The most critical parameter for calculating throughput on any high speed network is the station turnaround time; the time it takes a station to respond after receiving a frame. Since the 8044 handles all of the commands and responses of the Information Transfer State in hardware, the turnaround time is much faster than handling it in software, hence a higher throughput.

8044's Flexible Mode

In the "NON-AUTO" mode or Flexible mode, the SIU does not recognize or respond to any commands, nor does it manage acknowledgements, which means the CPU must handle link access, command recognition/response, acknowledgements and error recovery by itself. The Flexible mode allows the 8044 to have extended address fields and extended control fields, thus providing HDLC support. In the Flexible mode the 8044 can operate as a primary station, since it can transmit without being polled.

Front End Communications Processor

The 8044 can also be used as an intelligent HDLC/SDLC front end for a microporcessor, capable of extensively off-loading link control functions for

230876-8

the microporcessor. In some applications the 8044 can even be used for communications preprocessing, in addition to data link control. For this type of design the 8044 would communicate to the Host CPU through a FIFO, or dual port RAM. A block diagram of this design is given in Figure 8. A tightly coupled interface between the 8044 and the Host CPU would be established. The Host CPU would give the 8044 high level commands and data which the 8044 would convert to HDLC/SDLC. This particular type of design would be most appropriate for a primary Station which is normally a micro, mini, or mainframe computer. Sophisticated secondary stations could also take advantage of this design.

Since the 8044 has ROM on chip, all the communications software is non-volatile. The 8044 primary station could down-line-load software to 8044 secondary stations. Once down-line-loading is implemented, software updates to the primary and secondary stations could be done very inexpensively. The only things which would remain fixed in ROM are the HDLC/SDLC communications software and the software interface to the HOST.

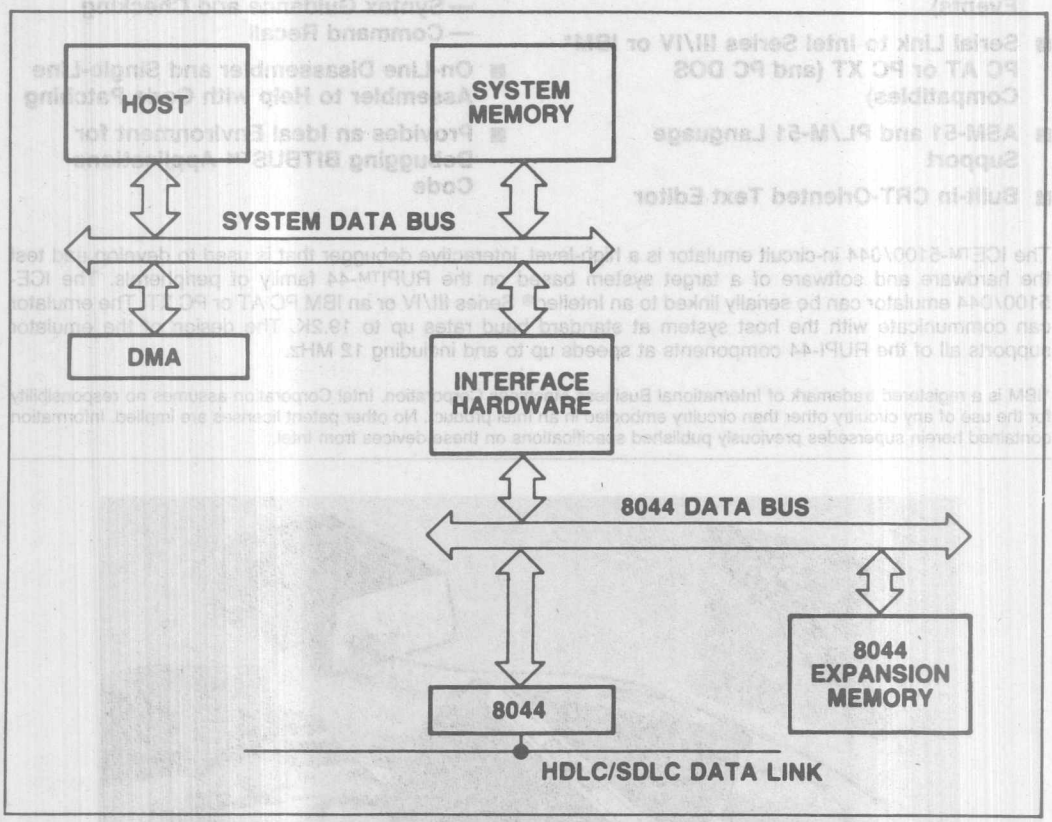


Figure 8. 8044 front end processor

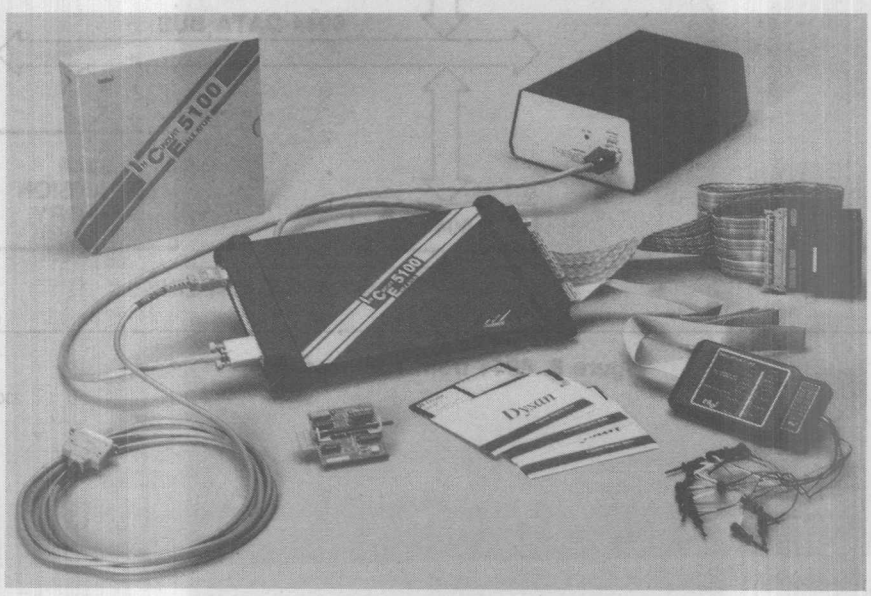
230876-9

ICE™-5100/044 In-Circuit Emulator for the RUPI™-44 Family

- **Precise, Full-Speed, Real-Time Emulation of the RUPI™-44 Family of Peripherals**
- **64 KB of Mappable High-Speed Emulation Memory**
- **254 24-bit Frames of Trace Memory (16 Bits Trace Program Execution Addresses and 8 Bits Trace Eternal Events)**
- **Serial Link to Intel Series III/IV or IBM* PC AT or PC XT (and PC DOS Compatibles)**
- **ASM-51 and PL/M-51 Language Support**
- **Built-in CRT-Oriented Text Editor**
- **Symbolic Debugging Enables Access to Memory Locations and Program Variables**
- **Four Address Breakpoints Plus In-Range, Out-of-Range, and Page Breaks**
- **Equipped with the Integrated Command Directory (ICD™) That Provides**
  - **— On-Line Help**
  - **— Syntax Guidance and Checking**
  - **— Command Recall**
- **On-Line Disassembler and Single-Line Assembler to Help with Code Patching**
- **Provides an Ideal Environment for Debugging BITBUS™ Applications Code**

The ICE™-5100/044 in-circuit emulator is a high-level, interactive debugger that is used to develop and test the hardware and software of a target system based on the RUPI™-44 family of peripherals. The ICE-5100/044 emulator can be serially linked to an Intellec® Series III/IV or an IBM PC AT or PC XT. The emulator can communicate with the host system at standard baud rates up to 19.2K. The design of the emulator supports all of the RUPI-44 components at speeds up to and including 12 MHz.

*IBM is a registered trademark of International Business Machines Corporation. Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel.



280325-1

November 1986
Order Number: 280325-001

PRODUCT OVERVIEW

The ICE-5100/044 emulator provides full emulation support for the RUPI-44 family of peripherals, including 8044-based BITBUS™ board products. The RUPI-44 family consists of the 8044, the 8744, and the 8344.

The ICE-5100/044 emulator enables hardware and software development to proceed simultaneously. With the ICE-5100/044, prototype hardware can be added to the system as it is designed and software can be developed prior to the completion of the hardware prototype. Software and hardware integration can occur while the product is being developed.

The ICE-5100/044 emulator assists four stages of development:

- Software debugging
- Hardware debugging
- System integration
- System test

Software Debugging

The ICE-5100/044 emulator can be operated without being connected to the target system and before any of the user's hardware is available (provided external data RAM is not needed). In this stand-alone mode, the ICE-5100/044 emulator can be used to facilitate program development.

Hardware Debugging

The ICE-5100/044 emulator's AC/DC parametric characteristics match the microcontroller's. The emulator's full-speed operation makes it a valuable tool for debugging hardware, including time-critical serial port, timer, and external interrupt interfaces.

System Integration

Integration of software and hardware can begin when the emulator is plugged into the microcontroller socket of the prototype system hardware. Hardware can be added, modified, and tested immediately. As each section of the user's hardware is completed, it can be added to the prototype. Thus, the hardware and software can be system tested in real-time operation as each section becomes available.

System Test

When the prototype is complete, it is tested with the final version of the system software. The ICE-5100/044 emulator is then used for real-time emulation of the microcontroller to debug the system as a completed unit.

The final product verfication test can be performed using the ROM or EPROM version of the microcontroller. Thus, the ICE-5100/044 emulator provides the ability to debug a prototype or production system at any stage in its development without introducing extraneous hardware or software test tools.

PHYSICAL DESCRIPTION

The ICE-5100/044 emulator consists of the following components (see Figure 1):

- Power supply
- AC and DC power cables
- Controller pod
- Serial Cable (host-specific)
- User probe assembly (consisting of the processor module and the user cable)
- Crystal power accessory (CPA)
- 40-pin target adaptor
- Clips assembly
- Software (includes the ICE-5100/044 emulator software, diagnostic software, and a tutorial)

The controller pod contains 64 KB of emulation memory, 254- by 24-bit frames of trace memory, and the control processor. In addition, the controller pod houses a BNC connector that can be used to connect up to 10 multi-ICE compatible emulators for synchronous starting and stopping of emulation.

The serial cable connects the host system to the controller pod. The serial cable supports a subset of the RS-232C signals.

The user probe assembly consists of a user cable and a processor module. The processor module houses the emulation processor and the interface logic. The target adaptor connects to the processor module and provides an electrical and mechanical interface to the target microcontroller socket.

The crystal power accessory (CPA) is a small, detachable board that connects to the controller pod and enables the ICE-5100/044 emulator to run in stand-alone mode. The target adaptor plugs into the socket on the CPA; the CPA then supplies clock and power to the user probe.

The clips assembly enables the user to trace external events. Eight bits of data are gathered on the rising edge of $\overline{PSEN}$ during opcode fetches. The clips information can be displayed using the CLIPS option with the PRINT command.

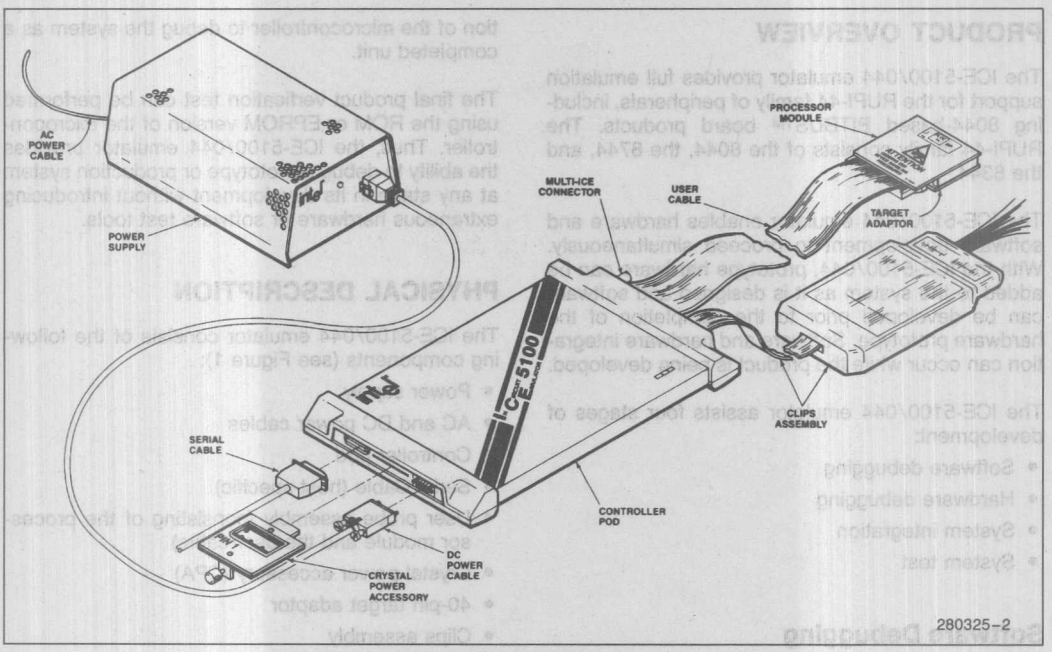


Figure 1. The ICE™-5100/044 Emulator Hardware

The ICE-5100-044 emulator software supports mnemonics, object file formats, and symbolic references generated by Intel's ASM-51 and PL/M-51 programming languages. Along with the ICE-5100/044 emulator software is a customer confidence test disk with diagnostic routines that check the operation of the hardware.

The on-line tutorial is written in the ICE-5100 command language. Thus, the user is able to interact with and use the ICE-5100/044 emulator while executing the tutorial.

A comprehensive set of documentation is provided with the ICE-5100/044 emulator.

ICE™-5100/044 EMULATOR FEATURES

The ICE-5100/044 emulator has been created to assist a product designer in developing, debugging and testing designs incorporating the RUPI-44 family of peripherals. The following sections detail some of the ICE-5100/044 emulator features.

Emulation

Emulation is the controlled execution of the user's software in the target hardware or in an artificial hardware environment that duplicates the microcontroller of the target system. Emulation is a transparent process that happens in real-time. The execution of the user software is facilitated with the ICE-5100/044 command language.

Memory Mapping

There is a 64 KB of memory that can be mapped to the CODE memory space in 4 KB blocks on 4 KB boundaries. By mapping memory to the ICE-5100/044 emulator, software development can proceed before the user hardware is available.

Memory Examination and Modification

The memory space for the 8044 microcontroller and its target hardware is fully accessible through the emulator. The ICE-5100/044 emulator refers to four physically distinct memory spaces, as follows:

- CODE—references program memory
- IDATA—references internal data memory
- RDATA—references special function register memory
- XDATA—references external data memory

ICE-5100/044 emulator commands that access memory use one of the special prefixes (e.g., CODE) to specify the memory space.

The microcontroller's special function registers and register bits can be accessed mnemonically (e.g., DPL, TCON, CY, P1.2) with the ICE-5100/044 emulator software.

Data can be displayed or modified in one of three bases: hexadecimal, decimal, or binary. Data can also be displayed or modified in one of two formats: ASCII or unsigned integer. Program code can be disassembled and displayed as ASM-51 assembler mnemonics. Code can be modified with standard ASM-51 statements using the built-in single-line assembler.

Symbolic references can be used to specify memory locations. A symbolic reference is a procedure name, line number, program variable, or label in the user program that corresponds to a location.

Some typical symbolic functions include:

- Changing or inspecting the value of a program variable by using its symbolic name to access the memory location.
- Defining break and trace events using symbolic references.
- Referencing variables as primitive data types. The primitive data types are ADDRESS, BIT, BOOLEAN, BYTE, CHAR (character), and WORD.

The ICE-5100/044 emulator maintains a virtual symbol table (VST) for program symbols. A maximum of 61 KB of host memory space is available for the VST. If the VST is larger than 61 KB, the excess is stored on available host system disk space and is paged in and out as needed. The size of the VST is limited only by the disk capacity of the host system.

Breakpoint Specifications

Breakpoints are used to halt a user program in order to examine the effect of the program's execution on the target system. The ICE-5100/044 emulator supports three different types of break specifications:

- Specific address break—specifying a single address point at which emulation is to be stopped.
- Range break—an arbitrary range of addresses can be specified to halt emulation. Program execution within or, optionally, outside the range halts emulation.
- Page break—up to 256 page breaks can be specified. A page break is defined as a range of addresses that is 256-bytes long and begins on a 256-byte address boundary.

Break registers are user-defined debug definitions used to create and store breakpoint definitions. Break registers can contain multiple breakpoint definitions and can optionally call debug procedures when emulation halts.

Trace Specifications

Tracing can be triggered using specifications similar to those used for breaking. Normally, the ICE-5100/044 emulator traces program activity while the user program is executing. With a trace specification, tracing can be triggered to occur only when specific conditions are met during execution. Up to 254 24-bit frames of trace information are collected in a buffer during emulation. Sixteen of the 24 bits trace instruction execution addresses, and 8 bits capture external events (CLIPS).

```
        /* Print newest four instructions in the buffer */
hlt>PRINT NEWEST 4
FRAME    ADDR      CODE      INSTRUCTIONS
(28)     300A      C02A      PUSH      2AH
(30      300C      2532      ADD       A, 32H
(32)     300E      F52A      MOV       2AH, A
(34)     3010      B53210    CJNE      A,32H, $+10H
hlt>
hlt>PRINT CLIPS OLDEST 2      /* Buffer display showing clips */
FRAME    ADDR      CODE      INSTRUCTIONS   CLIPS     (76543210)
(00)     007AH     0508      INC  INDX PTR             10101111
(01)     007CH     80E6      SJMP (#28)                00100010
```

280325-3

Figure 2. Selected Trace Buffer Displays

The trace buffer display is similar to an ASM-51 program listing as shown in Figure 2. The PRINT command enables the user to selectively display the contents of the trace buffer. The user has the option of displaying the clips information as well as dissassembled instructions.

Procedures

Debugging procedures (PROCs) are a user-defined group of ICE-5100/044 commands that are executed as one command. PROCs enable the user to define several commands in a named block structure. The commands are executed by entering thename of the PROC. The PROC bodies are a simple DO . . . END construct.

```
hlt>GO
FROM     ARM  FOREVER   TIL  USING     TRACE     ;     <execute>

hlt>GO FROM
<expr>

hlt>GO FROM 13H
<operator>     ARM  FOREVER   TIL  USING     TRACE     ;     <execute>

hlt>GO FROM 13H USING
BRKREG<brkreg name>

hlt>GO FROM 13H USING brl
,   TRACE     ;     <execute>

hlt>GO FROM 13H USING brl TRACE
<expr>   OUTSIDE   PAGE FROM TIL  <trcreg name>  ;     <execute>

hlt>GO FROM 13H USING brl TRACE traceit
;   <execute>
```

280325-4

Figure 3. The Integrated Command Directory for the GO Command

PROCs can simulate missing hardware or software, set breakpoints, collect debug information, and execute high-level software patches. PROCs can be copied to text files on disk, then recalled for use in later test sessions. PROCs can also serve as program diagnostics, implementing ICE-5100/044 emulator commands or user-defined definitions for special purposes.

On-Line Syntax Menu

A special syntax menu, called the Integrated Command directory (ICD), similar to the one used for the I²ICE™ system and the VLSiCE-96 emulator, aids in creating syntactically correct command lines. Figure 3 shows an example of the ICD and how it changes to reflect the options available for the GO command.

Help

The HELP command provides ICE-5100/044 emulation command assistance via the host system terminal. On-line HELP is available for the ICE-5100/044 emulator commands shown in Figure 4.

BITBUS™ Applications Support

The ICE-5100/044 emulator provides an ideal environment for developing applications code for BITBUS board products such as the RCB-44/10, the RCB-44/20, the PCX-344, and the iSBX™-344 board.

The BITBUS firmware, available separately as BITWARE, can be loaded into the ICE-5100/044 emulator's memory along with the user's code to enable rapid debug of 8044 BITBUS applications code.

DESIGN CONSIDERATIONS

The height of the processor module and the target adaptor need to be considered for target systems. Allow at least 1½ inches (3.8 cm) of space to fit the processor module and target adaptor. Figure 5 shows the dimensions of the processor module.

Execution of user programs that contain interrupt routines causes incorrect data to be stored in the trace buffer. When an interrupt occurs, the next instruction to be executed is placed into the trace buffer before it is actually executed. Following completion of the interrupt routine, the instruction is executed and again placed into the trace buffer.

ELECTRICAL CONSIDERATIONS

The emulation processor's user-pin timings and loadings are identical to the 8044 component, except as follows.

- Up to 25 pF of additional pin capacitance is contributed by the processor module and target adaptor assemblies.
- Pin 31, $\overline{EA}$, has approximately 32 pF of additional capacitance loading due to sensing circuitry.
- Pins 18 and 19, XTAL1 and XTAL2 respectively, have approximately 15-16 pF of additional capacitance when configured for crystal operation.

```
hlt>HELP

HELP is available for:

ADDRESS    APPEND   ASM      BASE      BIT        BOOLEAN     BRKREG
BYTE       CHAR     CI       CNTL_C    COMMENTS   CONSTRUCTS  COUNT
CURHOME    CURX     CURY     DCI       DEBUG      DEFINE      DIR
DISPLAY    DO       DYNASCOPE EDIT     ERROR      EVAL        EXIT
EXPRESSION GO       HELP     IF        INCLUDE    INVOCATION  ISTEP
KEYS       LABEL    LINES    LIST      LITERALLY  LOAD        LSTEP
MAP        MENU     MODIFY   MODULE    MSPACE     MTYPE       NAMESCOPE
OPERATOR   PAGING   PARTITION PRINT     PROC       PSEUDO_VAR  PUT
REFERENCE  REGS     REMOVE   REPEAT    RESET      RETURN      SAVE
STRING     SYMBOLIC SYNCSTART TEMPCHECK TRCREG    TYPES       VARIABLE
VERIFY     VERSION  WAIT     WORD      WRITE
hlt>
```

280325-5

Figure 4. HELP Menu

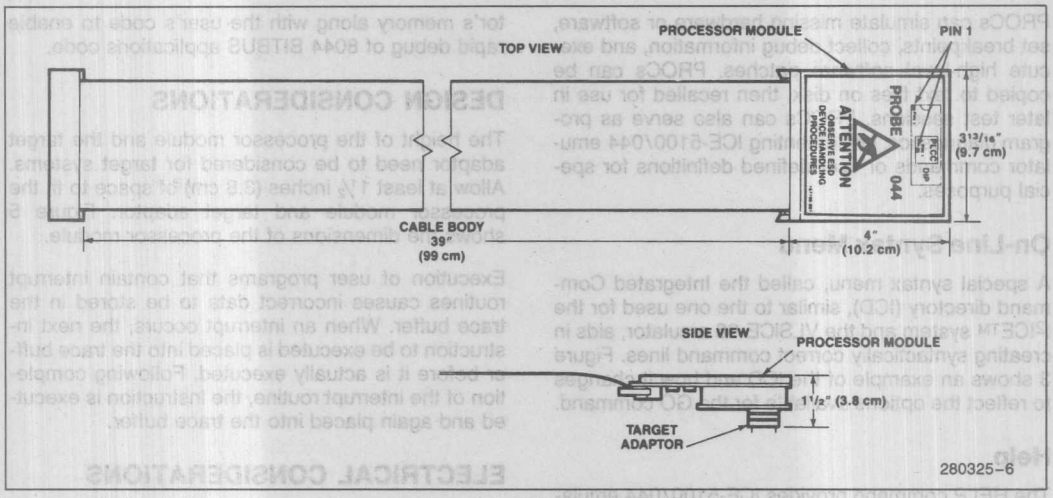


Figure 5. Processor Module Dimensions

HOST REQUIREMENTS

- IBM PC AT or PC XT (or PC DOS compatible) with 512 KB of available RAM and a hard disk running under the DOS 3.0 (or later) operating system.
- Intellec Series III/IV microcomputer development system running the ISIS or iNDX operating system respectively, with at least 512 KB of application memory resident.
- Disk drives—dual floppy or one hard disk and one floppy drive required.

ICE™-5100/044 EMULATOR SOFTWARE PACKAGE

- ICE-5100/044 emulator software
- ICE-5100/044 confidence tests
- ICE-5100 tutorial software

EMULATOR PERFORMANCE

Memory

Mappable full-speed emulation code memory	64 KB	Mappable to user or ICE-5100/044 emulator memory in 4 KB blocks on 4 KB boundaries.
Trace memory		254 x 24 bit frames
Virtual Symbol Table		A maximum of 61 KB of host memory space is available for the virtual symbol table (VST). The rest of the VST resides on disk and is paged in and out as needed.

PHYSICAL CHARACTERISTICS

Controller Pod

Width:	8-1/4"	(21 cm)
Height:	1-1/2"	(3.8 cm)
Depth:	13-1/2"	(34.3 cm)
Weight:	4 lbs	(1.85 kg)

User Cable

The user cable is 3 feet (approximately 1 m)

Processor Module

(With the target adaptor attached)

Width:	3-13/16"	(9.7 cm)
Height:	4"	(10.2 cm)
Depth:	1-1/2"	(3.8 cm)

Power Supply

Width:	7-5/8"	(18.1 cm)
Height:	4"	(10.06 cm)
Depth:	11"	(27.97 cm)
Weight	15 lbs	(6.1 kg)

Serial Cable

The serial cable is 12 feet (3.6 m).

ELECTRICAL CHARACTERISTICS

Power Supply

100-120V or 200-240V (selectable)
50-60 Hz
2 amps (AC max) @ 120V
1 amp (AC max) @ 240V

ENVIRONMENTAL CHARACTERISTICS

Operating temperature	+10°C to +40°C (50°F to 104°F)
Operating humidity	Maximum of 85% relative humidity, non-condensing

ORDERING INFORMATION

Emulator Hardware and Software

Order Code	Description
I044KITAD	This kit contains the ICE-5100/044 user probe assembly, power supply and cables, serial cables, target adaptor, CPA, ICE-5100 controller pod, software, and documentation for use with an IBM PC AT or PC XT. The kit also includes the 8051 Software Development Package and the AEDIT text editor for use on DOS systems. [Requires software license.]
I044KITD	This kit is the same as the I044KITAD excluding the 8051 Software Development Package and the AEDIT text editor. [Requires software license.]
I044KITAS	This kit contains the ICE-5100/044 user probe assembly, power supply and cables, serial cables, target adaptor, CPA, ICE-5100 controller pod, software, and documentation for use with Intel hosts (Series III/IV). The kit also includes the 8051 Software Development Package and the AEDIT text editor for use on the Series III/IV. [Requires software license.]
I044KITS	This kit is the same as the I044KITAS excluding the 8051 Software Development Package and the AEDIT text editor. [Requires software license.]

Software Only

Order Code	Description
SA044D	This kit contains the host, probe, diagnostic, and tutorial software on 5¼″ disks for use on an IBM PC AT or PC XT (requires DOS 3.0 or later). [Requires software license.]
SA044S	This kit contains the host, probe, diagnostic, and tutorial software on 8″ disks (both single-density and double-density) for use on a Series III, and on 5-¼″ disks for use on a Series IV. [Requires software license.]

Other Useful Intel® MCS®-51 Debug and Development Support Products

Order Code	Description
D86ASM51	**8051 Software Development Package** (DOS version)—Consists of the ASM-51 macro assembler which gives symbolic access to 8051 hardware features; the RL51 linker and relocator program that links modules generated by ASM-51; CONV51 which enables software written for the MCS-48 family to be up-graded to run on the 8051, and the LIB51 Librarian which programmers can use to create and maintain libraries of software object modules. Use with the DOS operating system (version 3.0 or later).
D86PLM51	**PL/M-51 Software Package** (DOS version)—Consists of the PL/M-51 compiler which provides high-level programming language support; the LIB51 utility that creates and maintains libraries of software object modules, and the RL51 linker and relocator program that links modules generated by ASM-51 and PL/M-51 and locates the linked object modules to absolute memory locations. Use with the DOS operating system (version 3.0 or later).
I86ASM51	**8051 Software Development Package** (ISIS version)—Same as the D86ASM51 package except this one is for use with the Series III.
I86PLM51	**PL/M-51 Sofware Package**—Same as the D86PLM51 package except this one is for use with the Series III and Series IV.
D86EDINL	AEDIT text editor for use with the DOS operating system.

Power Supply

100-120V or 200-240V (selectable)
50-60 Hz
2 amps (AC max) @ 120V
1 amp (AC max) @ 240V

ENVIRONMENTAL CHARACTERISTICS

Operating temperature	+ 10°C to + 40°C (50°F to 104°F)
Operating humidity	Maximum of 85% relative humidity, non-condensing

ORDERING INFORMATION

Emulator Hardware and Software

Order Code	Description
104AKITAD	This kit contains the ICE-5100/044 user probe assembly, power supply and cables, serial cables, target adapter, CPA, ICE-5100 controller pod, software, and documentation for use with an IBM PC AT or PC XT. The kit also includes the 8051 Software Development Package and the AEDIT text editor for use on DOS systems. [Requires software licenses.]
104AKITD	This kit is the same as the 104AKITAD excluding the 8051 Software Development Package and the AEDIT text editor. [Requires software licenses.]
104AKITAS	This kit contains the ICE-5100/044 user probe assembly, power supply and cables, serial cables, target adapter, CPA, ICE-5100 controller pod, software, and documentation for use with Intel hosts (Series III/IV). The kit also includes the 8051 Software Development Package and the AEDIT text editor for use on the Series III/IV. [Requires software licenses.]
104AKITS	This kit is the same as the 104AKITAS excluding the 8051 Software Development Package and the AEDIT text editor. [Requires software licenses.]

Order Code	Description
SA044D	This kit contains the host, probe, diagnostic, and tutorial software on 5¼" disks for use on IBM PC AT or PC XT (requires DOS 3.0 or later). [Requires software licenses.]
SA044S	This kit contains the host, probe, diagnostic, and tutorial software on 8" (both single-density and double-density) for use on a Series III, and on 5-¼" disks for use on a Series IV. [Requires software licenses.]

Other Useful Intel MCS®-51 Debug and Development Support Products

Order Code	Description
D86ASM51	8051 Software Development Package (DOS version)—Consists of the ASM-51 macro assembler which gives symbolic access to 8051 hardware features; the RL51 linker and relocator program that links modules generated by ASM-51; CONV51 which enables software written for the MCS-48 family to be up-graded to run on the 8051; and the LIB51 Librarian which programmers can use to create and maintain libraries of software object modules. Use with the DOS operating system (version 3.0 or later).
D86PLM51	PL/M-51 Software Package (DOS version)—Consists of the PL/M-51 compiler which provides high-level programming language support; the LIB51 utility that creates and maintains libraries of software object modules; and the RL51 linker and relocator program that links modules generated by ASM-51 and PL/M-51 and locates the linked object modules to absolute memory locations. Use with the DOS operating system (version 3.0 or later).
I86ASM51	8051 Software Development Package (ISIS version)—Same as the D86ASM51 package except this one is for use with the Series III.
I86PLM51	PL/M-51 Software Package—Same as the D86PLM51 package except this one is for use with the Series III and Series IV.
D86EDINL	AEDIT text editor for use with the DOS operating system.

8080A/8080A-1/8080A-2
8-BIT N-CHANNEL MICROPROCESSOR

- **TTL Drive Capability**
- **2 μs (−1:1.3 μs, −2:1.5 μs) Instruction Cycle**
- **Powerful Problem Solving Instruction Set**
- **6 General Purpose Registers and an Accumulator**
- **16-Bit Program Counter for Directly Addressing up to 64K Bytes of Memory**
- **16-Bit Stack Pointer and Stack Manipulation Instructions for Rapid Switching of the Program Environment**
- **Decimal, Binary, and Double Precision Arithmetic**
- **Ability to Provide Priority Vectored Interrupts**
- **512 Directly Addressed I/O Ports**
- **Available in EXPRESS — Standard Temperature Range**
- **Available in 40-Lead Cerdip and Plastic Packages**
  (See Packaging Spec, Order #231369)

The Intel® 8080A is a complete 8-bit parallel central processing unit (CPU). It is fabricated on a single LSI chip using Intel's n-channel silicon gate MOS process. This offers the user a high performance solution to control and processing applications.

The 8080A contains 6 8-bit general purpose working registers and an accumulator. The 6 general purpose registers may be addressed individually or in pairs providing both single and double precision operators. Arithmetic and logical instructions set or reset 4 testable flags. A fifth flag provides decimal arithmetic operation.

The 8080A has an external stack feature wherein any portion of memory may be used as a last in/first out stack to store/retrieve the contents of the accumulator, flags, program counter, and all of the 6 general purpose registers. The 16-bit stack pointer controls the addressing of this external stack. This stack gives the 8080A the ability to easily handle multiple level priority interrupts by rapidly storing and restoring processor status. It also provides almost unlimited subroutine nesting.

This microprocessor has been designed to simplify systems design. Separate 16-line address and 8-line bidirectional data busses are used to facilitate easy interface to memory and I/O. Signals to control the interface to memory and I/O are provided directly by the 8080A. Ultimate control of the address and data busses resides with the HOLD signal. It provides the ability to suspend processor operation and force the address and data busses into a high impedance state. This permits OR-tying these busses with other controlling devices for (DMA) direct memory access or multi-processor operation.

NOTE:
The 8080A is functionally and electrically compatible with the Intel 8080.

November 1986
Order Number: 231453-001

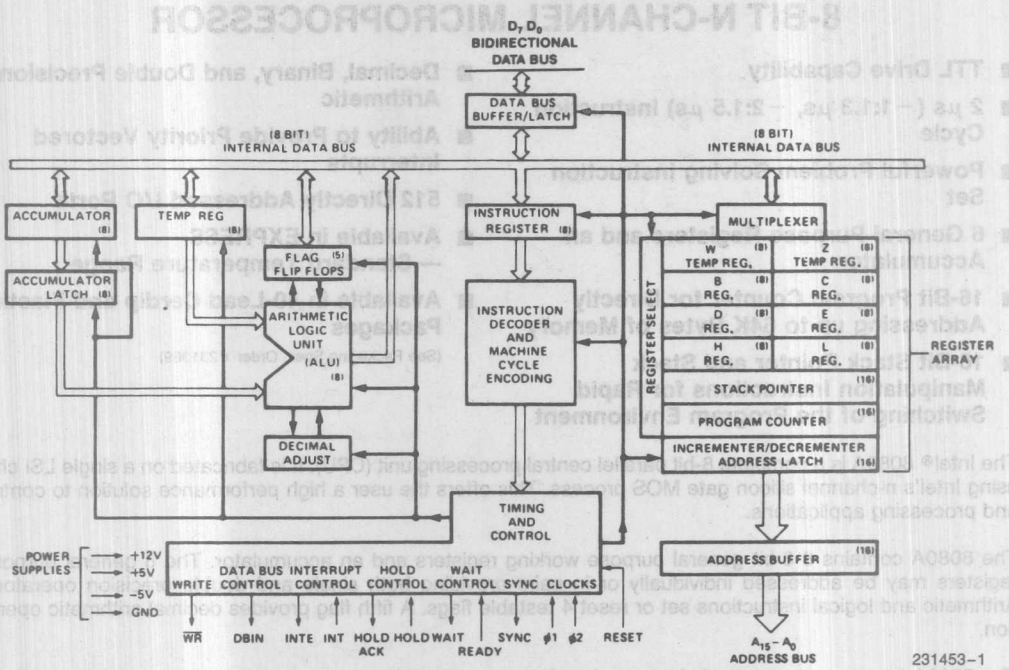


Figure 1. Block Diagram

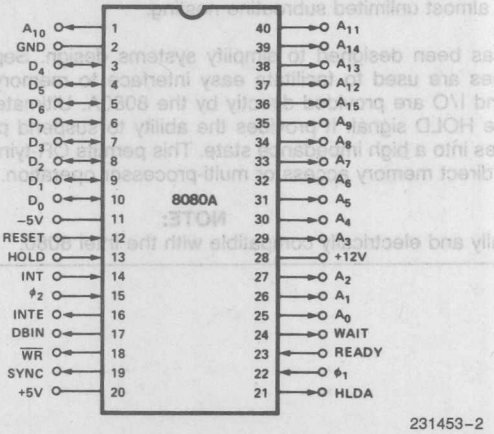


Figure 2. Pin Configuration

Table 1. Pin Description

Symbol	Type	Name and Function
A_{15}–A_0	O	**ADDRESS BUS:** The address bus provides the address to memory (up to 64K 8-bit words) or denotes the I/O device number for up to 256 input and 256 output devices. A_0 is the least significant address bit.
D_7–D_0	I/O	**DATA BUS:** The data bus provides bi-directional communication between the CPU, memory, and I/O devices for instructions and data transfers. Also, during the first clock cycle of each machine cycle, the 8080A outputs a status word on the data bus that describes the current machine cycle. D_0 is the least significant bit.
SYNC	O	**SYNCHRONIZING SIGNAL:** The SYNC pin provides a signal to indicate the beginning of each machine cycle.
DBIN	O	**DATA BUS IN:** The DBIN signal indicates to external circuits that the data bus is in the input mode. This signal should be used to enable the gating of data onto the 8080A data bus from memory or I/O.
READY	I	**READY:** The READY signal indicates to the 8080A that valid memory or input data is available on the 8080A data bus. This signal is used to synchronize the CPU with slower memory or I/O devices. If after sending an address out the 8080A does not receive a READY input, the 8080A will enter a WAIT state for as long as the READY line is low. READY can also be used to single step the CPU.
WAIT	O	**WAIT:** The WAIT signal acknowledges that the CPU is in a WAIT state.
$\overline{WR}$	O	**WRITE:** The $\overline{WR}$ signal is used for memory WRITE or I/O output control. The data on the data bus is stable while the $\overline{WR}$ signal is active low ($\overline{WR} = 0$).
HOLD	I	**HOLD:** The HOLD signal requests the CPU to enter the HOLD state. The HOLD state allows an external device to gain control of the 8080A address and data bus as soon as the 8080A has completed its use of these busses for the current machine cycle. It is recognized under the following conditions: • the CPU is in the HALT state. • the CPU is in the T2 or TW state and the READY signal is active. As a result of entering the HOLD state the CPU ADDRESS BUS (A_{15}–A_0) and DATA BUS (D_7–D_0) will be in their high impedance state. The CPU acknowledges its state with the HOLD ACKNOWLEDGE (HLDA) pin.
HLDA	O	**HOLD ACKNOWLEDGE:** The HLDA signal appears in response to the HOLD signal and indicates that the data and address bus will go to the high impedance state. The HLDA signal begins at: • T3 for READ memory or input. • The Clock Period following T3 for WRITE memory or OUTPUT operation. In either case, the HLDA signal appears after the rising edge of ϕ_2.
INTE	O	**INTERRUPT ENABLE:** Indicates the content of the internal interrupt enable flip/flop. This flip/flop may be set or reset by the Enable and Disable Interrupt instructions and inhibits interrupts from being accepted by the CPU when it is reset. It is automatically reset (disabling further interrupts) at time T1 of the instruction fetch cycle (M1) when an interrupt is accepted and is also reset by the RESET signal.
INT	I	**INTERRUPT REQUEST:** The CPU recognizes an interrupt request on this line at the end of the current instruction or while halted. If the CPU is in the HOLD state or if the Interrupt Enable flip/flop is reset it will not honor the request.
RESET[1]	I	**RESET:** While the RESET signal is activated, the content of the program counter is cleared. After RESET, the program will start at location 0 in memory. The INTE and HLDA flip/flops are also reset. Note that the flags, accumulator, stack pointer, and registers are not cleared.
V_{SS}		**GROUND:** Reference.
V_{DD}		**POWER:** +12 ±5% V.
V_{CC}		**POWER:** +5 ±5% V.
V_{BB}		**POWER:** −5 ±5% V.
ϕ_1, ϕ_2		**CLOCK PHASES:** 2 externally supplied clock phases. (non TTL compatible)

NOTE:
1. The RESET signal must be active for a minimum of 3 clock cycles.

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... 0°C to +70°C
Storage Temperature ..... −65°C to +150°C
All Input or Output Voltages with Respect to V_{BB} ..... −0.3V to +20V
V_{CC}, V_{DD} and V_{SS} with Respect to V_{BB} ..... −0.3V to +20V
Power Dissipation 1.5W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

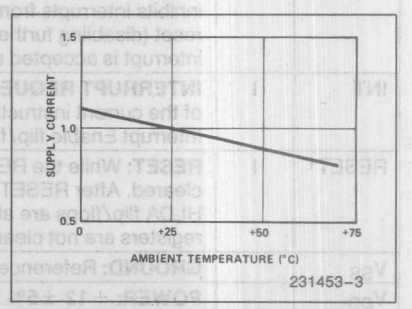
T_A = 0°C to 70°C, V_{DD} = +12V ±5%, V_{CC} = +5V ±5%, V_{BB} = −5V ±5%, V_{SS} = 0V; unless otherwise noted

Symbol	Parameter	Min	Typ	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	V_{SS} − 1		V_{SS} + 0.8	V	
V_{IHC}	Clock Input High Voltage	9.0		V_{DD} + 1	V	
V_{IL}	Input Low Voltage	V_{SS} − 1		V_{SS} + 0.8	V	
V_{IH}	Input High Voltage	3.3		V_{CC} + 1	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 1.9 mA on All Outputs, I_{OH} = −150 μA.
V_{OH}	Output High Voltage	3.7			V	
$I_{DD\ (AV)}$	Avg. Power Supply Current (V_{DD})		40	70	mA	Operation T_{CY} = 0.48 μs
$I_{CC\ (AV)}$	Avg. Power Supply Current (V_{CC})		60	80	mA	
$I_{BB\ (AV)}$	Avg. Power Supply Current (V_{BB})		0.01	1	mA	
I_{IL}	Input Leakage			±10	μA	$V_{SS} \le V_{IN} \le V_{CC}$
I_{CL}	Clock Leakage			±10	μA	$V_{SS} \le V_{CLOCK} \le V_{DD}$
I_{DL}	Data Bus Leakage in Input Mode			−100 −2.0	μA mA	$V_{SS} \le V_{IN} \le V_{SS}$ + 0.8V V_{SS} + 0.8V $\le V_{IN} \le V_{CC}$
I_{FL}	Address and Data Bus Leakage During HOLD			+10 −100	μA	$V_{ADDR/DATA} = V_{CC}$ $V_{ADDR/DATA} = V_{SS}$ + 0.45V

CAPACITANCE

T_A = 25°C, V_{CC} = V_{DD} = V_{SS} = 0V, V_{BB} = −5V

Symbol	Parameter	Typ	Max	Unit	Test Condition
C_ϕ	Clock Capacitance	17	25	pF	f_c = 1 MHz
C_{IN}	Input Capacitance	6	10	pF	Unmeasured Pins
C_{OUT}	Output Capacitance	10	20	pF	Returned to V_{SS}

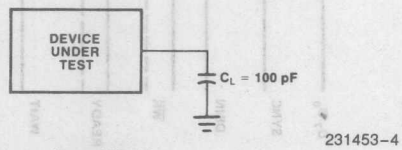


Typical Supply Current vs Temperature, Normalized
ΔI Supply/ΔT_A = −0.45%/°C

A.C. CHARACTERISTICS (8080A) T_A = 0°C to 70°C, V_{DD} = +12V ±5%, V_{CC} = +5V ±5%, V_{BB} = −5V ±5%, V_{SS} = 0V; unless otherwise noted

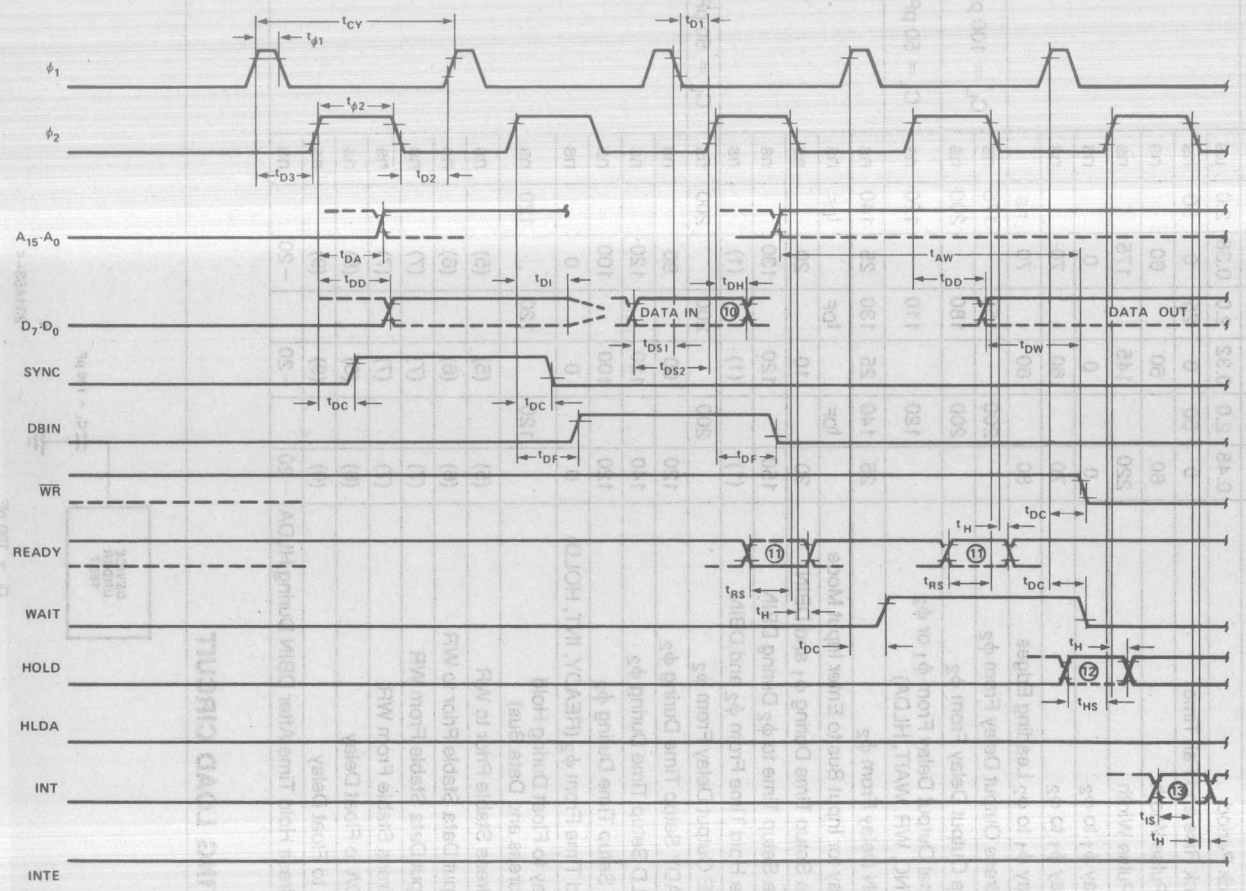
Symbol	Parameter	Min	Max	−1 Min	−1 Max	−2 Min	−2 Max	Unit	Test Condition
t_{CY}(3)	Clock Period	0.48	2.0	0.32	2.0	0.38	2.0	μs	
t_r, t_f	Clock Rise and Fall Time	0	50	0	25	0	50	ns	
$t_{\phi 1}$	ϕ_1 Pulse Width	60		50		60		ns	
$t_{\phi 2}$	ϕ_2 Pulse Width	220		145		175		ns	
t_{D1}	Delay ϕ_1 to ϕ_2	0		0		0		ns	
t_{D2}	Delay ϕ_1 to ϕ_2	70		60		70		ns	
t_{D3}	Delay ϕ_1 to ϕ_2 Leading Edges	80		60		70	ns		
t_{DA}	Address Output Delay From ϕ_2		200		150		175	ns	C_L = 100 pF
t_{DD}	Data Output Delay From ϕ_2		200		180		200	ns	
t_{DC}	Signal Output Delay From ϕ_1 or ϕ_2 (SYNC, WR, WAIT, HLDA)		120		110		120	ns	C_L = 50 pF
t_{DF}	DBIN Delay From ϕ_2	25	140	25	130	25	140	ns	
t_{DI}(1)	Delay for Input Bus to Enter Input Mode		t_{DF}		t_{DF}		t_{DF}	ns	
t_{DS1}	Data Setup Time During ϕ_1 and DBIN	30		10		20		ns	
t_{DS2}	Data Setup Time to ϕ_2 During DBIN	150		120		130		ns	
t_{DH}(1)	Data Hold Time From ϕ_2 and DBIN	(1)		(1)		(1)		ns	
t_{IE}	INTE Output Delay From ϕ_2		200		200		200	ns	C_L = 50 pF
t_{RS}	READY Setup Time During ϕ_2	120		90		90		ns	
t_{HS}	HOLD Setup Time During ϕ_2	140		120		120		ns	
t_{IS}	INT Setup Time During ϕ_2	120		100		100		ns	
t_H	Hold Time From ϕ_2 (READY, INT, HOLD)	0		0		0		ns	
t_{FD}	Delay to Float During Hold (Address and Data Bus)		120		120		120	ns	
t_{AW}	Address Stable Prior to WR	(5)		(5)		(5)		ns	
t_{DW}	Output Data Stable Prior to WR	(6)		(6)		(6)		ns	
t_{WD}	Output Data Stable From WR	(7)		(7)		(7)		ns	
t_{WA}	Address Stable From WR	(7)		(7)		(7)		ns	
t_{HF}	HLDA to Float Delay	(8)		(8)		(8)		ns	
t_{WF}	WR to Float Delay	(9)		(9)		(9)		ns	
t_{AH}	Address Hold Time After DBIN During HLDA	−20		−20		−20		ns	

A.C. TESTING LOAD CIRCUIT



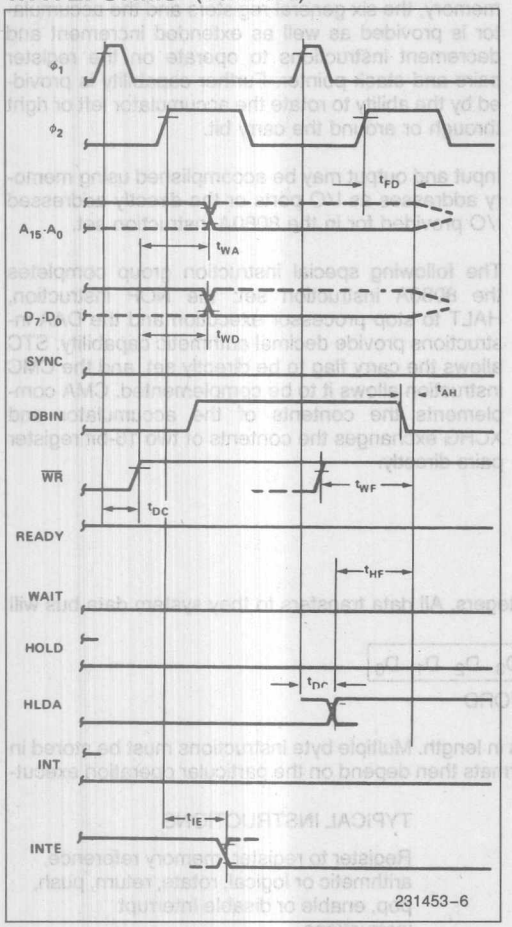
C_L = 100 pF
C_L Includes Jig Capacitance

WAVEFORMS



NOTE:
Timing measurements are made at the following reference voltages: CLOCK "1" = 8.0V, "0" = 1.0V; INPUTS "1" = 3.3V, "0" = 0.8V; OUTPUTS "1" = 2.0V, "0" = 0.8V

WAVEFORMS (Continued)

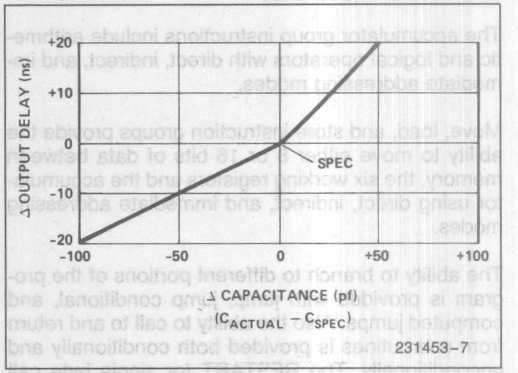


NOTES:

(Parenthesis gives −1, −2 specifications, respectively.)

1. Data input should be enabled with DBIN status. No bus conflict can then occur and data hold time is assured.
   t_{DH} = 50 ns or t_{DF}, whichever is less.
2. $t_{CY} = t_{D3} + t_{r\phi2} + t_{\phi2} + t_{f\phi2} + t_{D2} + t_{r\phi1} \geq$ 480 ns (−1:320 ns, − 2:380 ns).

Typical Δ Output Delay vs Δ Capacitance



3. The following are relevant when interfacing the 8080A to devices having V_{IH} = 3.3V:
   a) Maximum output rise time from 0.8V to 3.3V = 100 ns @ C_L = SPEC.
   b) Output delay when measured to 3.0V = SPEC +60 ns @ C_L = SPEC.
   c) If C_L = SPEC, add 0.6 ns/pF if $C_L > C_{SPEC}$, subtract 0.3 ns/pF (from modified delay) if $C_L < C_{SPEC}$.
4. $t_{AW} = 2\ t_{CY} - t_{D3} - t_{r\phi2} - 140$ ns (−1:110 ns, − 2:130 ns).
5. $t_{DW} = t_{CY} - t_{D3} - t_{r\phi2} - 170$ ns (−1:150 ns, − 2:170 ns).
6. If not HLDA, $t_{WD} = t_{WA} = t_{D3} + t_{r\phi2} + 10$ ns. If HLDA, $t_{WD} = t_{WA} = t_{WF}$.
7. $t_{HF} = t_{D3} + t_{r\phi2} - 50$ ns.
8. $t_{WF} = t_{D3} + t_{r\phi2} - 10$ ns.
9. Data in must be stable for this period during DBIN T_3. Both t_{DS1} and t_{DS2} must be satisfied.
10. Ready signal must be stable for this period during T_2 or T_W. (Must be externally synchronized.)
11. Hold signal must be stable for this period during T_2 or T_W when entering hold mode, and during T_3, T_4, T_5 and T_{WH} when in hold mode. (External synchronization is not required.)
12. Interrupt signal must be stable during this period of the last clock cycle of any instruction in order to be recognized on the following instruction. (External synchronization is not required.)
13. This timing diagram shows timing relationships only; it does not represent any specific machine cycle.

INSTRUCTION SET

The accumulator group instructions include arithmetic and logical operators with direct, indirect, and immediate addressing modes.

Move, load, and store instruction groups provide the ability to move either 8 or 16 bits of data between memory, the six working registers and the accumulator using direct, indirect, and immediate addressing modes.

The ability to branch to different portions of the program is provided with jump, jump conditional, and computed jumps. Also the ability to call to and return from subroutines is provided both conditionally and unconditionally. The RESTART (or single byte call instruction) is useful for interrupt vector operation.

Double precision operators such as stack manipulation and double add instructions extend both the arithmetic and interrupt handling capability of the 8080A. The ability to increment and decrement memory, the six general registers and the accumulator is provided as well as extended increment and decrement instructions to operate on the register pairs and stack pointer. Further capability is provided by the ability to rotate the accumulator left or right through or around the carry bit.

Input and output may be accomplished using memory addresses as I/O ports or the directly addressed I/O provided for in the 8080A instruction set.

The following special instruction group completes the 8080A instruction set: the NOP instruction, HALT to stop processor execution and the DAA instructions provide decimal arithmetic capability. STC allows the carry flag to be directly set, and the CMC instruction allows it to be complemented. CMA complements the contents of the accumulator and XCHG exchanges the contents of two 16-bit register pairs directly.

Data and Instruction Formats

Data in the 8080A is stored in the form of 8-bit binary integers. All data transfers to they system data bus will be in the same format.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

DATA WORD

The program instructions may be one, two, or three bytes in length. Multiple byte instructions must be stored in successive words in program memory. The instruction formats then depend on the particular operation executed.

		TYPICAL INSTRUCTIONS
One Byte Instructions		
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	OP CODE	Register to register, memory reference, arithmetic or logical, rotate, return, push, pop, enable or disable Interrupt instructions
Two Byte Instructions		
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	OP CODE	Immediate mode or I/O instructions
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	OPERAND	
Three Byte Instructions		
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	OP CODE	Jump, call or direct load and store instructions
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	LOW ADDRESS OR OPERAND 1	
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	HIGH ADDRESS OR OPERAND 2	

For the 8080A a logic "1" is defined as a high level and a logic "0" is defined as a low level.

Table 2. Instruction Set Summary

Mnemonic*	Instruction Code (1) D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Operations Description	Clock Cycles (2)
MOVE, LOAD, AND STORE			
MOVr1,r2	0 1 D D D S S S	Move register to register	5
MOV M,r	0 1 1 1 0 S S S	Move register to memory	7
MOV r,M	0 1 D D D 1 1 0	Move memory to register	7
MVI r	0 0 D D D 1 1 0	Move immediate register	7
MVI M	0 0 1 1 0 1 1 0	Move immediate memory	10
LXI B	0 0 0 0 0 0 0 1	Load immediate register Pair B & C	10
LXI D	0 0 0 1 0 0 0 1	Load immediate register Pair D & E	10
LXI H	0 0 1 0 0 0 0 1	Load immediate register Pair H & L	10
STAX B	0 0 0 0 0 0 1 0	Store A indirect	7
STAX D	0 0 0 1 0 0 1 0	Store A indirect	7
LDAX B	0 0 0 0 1 0 1 0	Load A indirect	7
LDAX D	0 0 0 1 1 0 1 0	Load A indirect	7
STA	0 0 1 1 0 0 1 0	Store A direct	13
LDA	0 0 1 1 1 0 1 0	Load A direct	13
SHLD	0 0 1 0 0 0 1 0	Store H & L direct	16
LHLD	0 0 1 0 1 0 1 0	Load H & L direct	16
XCHG	1 1 1 0 1 0 1 1	Exchange D & E, H & L Registers	4
STACK OPS			
PUSH B	1 1 0 0 0 1 0 1	Push register Pair B & C on stack	11
PUSH D	1 1 0 1 0 1 0 1	Push register Pair D & E on stack	11
PUSH H	1 1 1 0 0 1 0 1	Push register Pair H & L on stack	11
PUSH PSW	1 1 1 1 0 1 0 1	Push A and Flags on stack	11
POP B	1 1 0 0 0 0 0 1	Pop register Pair B & C off stack	10
POP D	1 1 0 1 0 0 0 1	Pop register Pair D & E off stack	10
POP H	1 1 1 0 0 0 0 1	Pop register Pair H & L off stack	10
POP PSW	1 1 1 1 0 0 0 1	Pop A and Flags off stack	10
XTHL	1 1 1 0 0 0 1 1	Exchange top of stack, H & L	18
SPHL	1 1 1 1 1 0 0 1	H & L to stack pointer	5
LXI SP	0 0 1 1 0 0 0 1	Load immediate stack pointer	10
INX SP	0 0 1 1 0 0 1 1	Increment stack pointer	5
DCX SP	0 0 1 1 1 0 1 1	Decrement stack pointer	5
JUMP			
JMP	1 1 0 0 0 0 1 1	Jump unconditional	10
JC	1 1 0 1 1 0 1 0	Jump on carry	10
JNC	1 1 0 1 0 0 1 0	Jump on no carry	10
JZ	1 1 0 0 1 0 1 0	Jump on zero	10
JNZ	1 1 0 0 0 0 1 0	Jump on no zero	10
JP	1 1 1 1 0 0 1 0	Jump on positive	10
JM	1 1 1 1 1 0 1 0	Jump on minus	10
JPE	1 1 1 0 1 0 1 0	Jump on parity even	10
JPO	1 1 1 0 0 0 1 0	Jump on parity odd	10
PCHL	1 1 1 0 1 0 0 1	H & L to program counter	5
CALL			
CALL	1 1 0 0 1 1 0 1	Call unconditional	17
CC	1 1 0 1 1 1 0 0	Call on carry	11/17
CNC	1 1 0 1 0 1 0 0	Call on no carry	11/17
CZ	1 1 0 0 1 1 0 0	Call on zero	11/17
CNZ	1 1 0 0 0 1 0 0	Call on no zero	11/17
CP	1 1 1 1 0 1 0 0	Call on positive	11/17
CM	1 1 1 1 1 1 0 0	Call on minus	11/17
CPE	1 1 1 0 1 1 0 0	Call on parity even	11/17
CPO	1 1 1 0 0 1 0 0	Call on parity odd	11/17
RETURN			
RET	1 1 0 0 1 0 0 1	Return	10
RC	1 1 0 1 1 0 0 0	Return on carry	5/11
RNC	1 1 0 1 0 0 0 0	Return on no carry	5/11
RZ	1 1 0 0 1 0 0 0	Return on zero	5/11
RNZ	1 1 0 0 0 0 0 0	Return on no zero	5/11
RP	1 1 1 1 0 0 0 0	Return on positive	5/11
RM	1 1 1 1 1 0 0 0	Return on minus	5/11
RPE	1 1 1 0 0 0 0 0	Return on parity even	5/11
RPO	1 1 1 0 0 0 0 0	Return on parity odd	5/11
RESTART			
RST	1 1 A A A 1 1 1	Restart	11
INCREMENT AND DECREMENT			
INR r	0 0 D D D 1 0 0	Increment register	5
DCR r	0 0 D D D 1 0 1	Decrement register	5
INR M	0 0 1 1 0 1 0 0	Increment memory	10
DCR M	0 0 1 1 0 1 0 1	Decrement memory	10
INX B	0 0 0 0 0 0 1 1	Increment B & C registers	5
INX D	0 0 0 1 0 0 1 1	Increment D & E registers	5
INX H	0 0 1 0 0 0 1 1	Increment H & L registers	5
DCX B	0 0 0 0 1 0 1 1	Decrement B & C	5
DCX D	0 0 0 1 1 0 1 1	Decrement D & E	5
DCX H	0 0 1 0 1 0 1 1	Decrement H & L	5
ADD			
ADD r	1 0 0 0 0 S S S	Add register to A	4
ADC r	1 0 0 0 1 S S S	Add register to A with carry	4
ADD M	1 0 0 0 0 1 1 0	Add memory to A	7
ADC M	1 0 0 0 1 1 1 0	Add memory to A with carry	7
ADI	1 1 0 0 0 1 1 0	Add immediate to A	7
ACI	1 1 0 0 1 1 1 0	Add immediate to A with carry	7
DAD B	0 0 0 0 1 0 0 1	Add B & C to H & L	10
DAD D	0 0 0 1 1 0 0 1	Add D & E to H & L	10
DAD H	0 0 1 0 1 0 0 1	Add H & L to H & L	10
DAD SP	0 0 1 1 1 0 0 1	Add stack pointer to H & L	10

Table 2. Instruction Set Summary (Continued)

Mnemonic*	Instruction Code (1) D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Operations Description	Clock Cycles (2)
SUBTRACT			
SUB r	1 0 0 1 0 S S S	Subtract register from A	4
SBB r	1 0 0 1 1 S S S	Subtract register from A with borrow	4
SUB M	1 0 0 1 0 1 1 0	Subtract memory from A	7
SBB M	1 0 0 1 1 1 1 0	Subtract memory from A with borrow	7
SUI	1 1 0 1 0 1 1 0	Subtract immediate from A	7
SBI	1 1 0 1 1 1 1 0	Subtract immediate from A with borrow	7
LOGICAL			
ANA r	1 0 1 0 0 S S S	And register with A	4
XRA r	1 0 1 0 1 S S S	Exclusive or register with A	4
ORA r	1 0 1 1 0 S S S	Or register with A	4
CMP r	1 0 1 1 1 S S S	Compare register with A	4
ANA M	1 0 1 0 0 1 1 0	And memory with A	7
XRA M	1 0 1 0 1 1 1 0	Exclusive Or memory with A	7
ORA M	1 0 1 1 0 1 1 0	Or memory with A	7
CMP M	1 0 1 1 1 1 1 0	Compare memory with A	7
ANI	1 1 1 0 0 1 1 0	And immediate with A	7
XRI	1 1 1 0 1 1 1 0	Exclusive Or immediate with A	7
ORI	1 1 1 1 0 1 1 0	Or immediate with A	7
CPI	1 1 1 1 1 1 1 0	Compare immediate with A	7

Mnemonic*	Instruction Code (1) D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Operations Description	Clock Cycles (2)
ROTATE			
RLC	0 0 0 0 0 1 1 1	Rotate A left	4
RRC	0 0 0 0 1 1 1 1	Rotate A right	4
RAL	0 0 0 1 0 1 1 1	Rotate A left through carry	4
RAR	0 0 0 1 1 1 1 1	Rotate A right through carry	4
SPECIALS			
CMA	0 0 1 0 1 1 1 1	Complement A	4
STC	0 0 1 1 0 1 1 1	Set carry	4
CMC	0 0 1 1 1 1 1 1	Complement carry	4
DAA	0 0 1 0 0 1 1 1	Decimal adjust A	4
INPUT/OUTPUT			
IN	1 1 0 1 1 0 1 1	Input	10
OUT	1 1 0 1 0 0 1 1	Output	10
CONTROL			
EI	1 1 1 1 1 0 1 1	Enable Interrupts	4
DI	1 1 1 1 0 0 1 1	Disable Interrupt	4
NOP	0 0 0 0 0 0 0 0	No-operation	4
HLT	0 1 1 1 0 1 1 0	Halt	7

NOTES:
1. DDD or SSS: B = 000, C = 001, D = 010, E = 011, H = 100, L = 101, Memory = 110, A = 111.
2. Two possible cycle times (6/12) indicate instruction cycles dependent on condition flags.

*All mnemonics copyright © Intel Corporation 1977

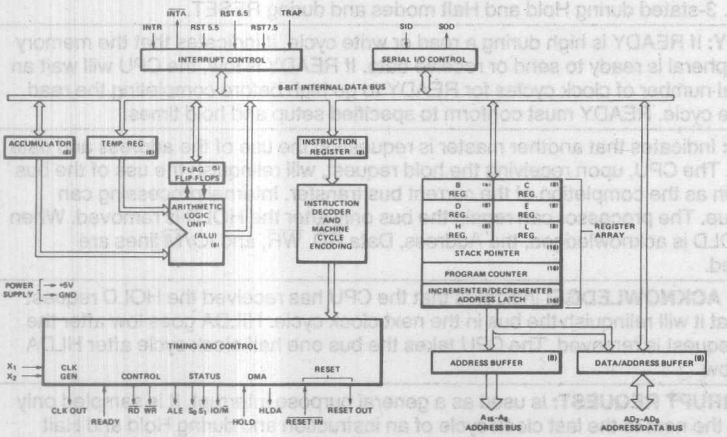
8085AH/8085AH-2/8085AH-1
8-BIT HMOS MICROPROCESSORS

- **Single +5V Power Supply with 10% Voltage Margins**
- **3 MHz, 5 MHz and 6 MHz Selections Available**
- **20% Lower Power Consumption than 8085A for 3 MHz and 5 MHz**
- **1.3 μs Instruction Cycle (8085AH); 0.8 μs (8085AH-2); 0.67 μs (8085AH-1)**
- **100% Software Compatible with 8080A**
- **On-Chip Clock Generator (with External Crystal, LC or RC Network)**
- **On-Chip System Controller; Advanced Cycle Status Information Available for Large System Control**
- **Four Vectored Interrupt Inputs (One Is Non-Maskable) Plus an 8080A-Compatible Interrupt**
- **Serial In/Serial Out Port**
- **Decimal, Binary and Double Precision Arithmetic**
- **Direct Addressing Capability to 64K Bytes of Memory**
- **Available in 40-Lead Cerdip and Plastic Packages**
  (See Packaging Spec., Order #231369)

The Intel 8085AH is a complete 8-bit parallel Central Processing Unit (CPU) implemented in N-channel, depletion load, silicon gate technology (HMOS). Its instruction set is 100% software compatible with the 8080A microprocessor, and it is designed to improve the present 8080A's performance by higher system speed. Its high level of system integration allows a minimum system of three IC's [8085AH (CPU), 8156H (RAM/IO) and 8755A (EPROM/IO)] while maintaining total system expandability. The 8085AH-2 and 8085AH-1 are faster versions of the 8085AH.

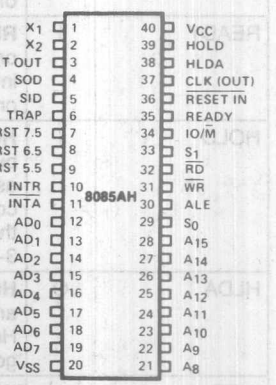
The 8085AH incorporates all of the features that the 8224 (clock generator) and 8228 (system controller) provided for the 8080A, thereby offering a higher level of system integration.

The 8085AH uses a multiplexed data bus. The address is split between the 8-bit address bus and the 8-bit data bus. The on-chip address latches of 8155H/8156H/8755A memory products allow a direct interface with the 8085AH.



231718-1

Figure 1. 8085AH CPU Functional Block Diagram



231718-2

Figure 2. 8085AH Pin Configuration

September 1987
Order Number: 231718-001

Table 1. Pin Description

Symbol	Type	Name and Function
A_8–A_{15}	O	**ADDRESS BUS:** The most significant 8 bits of memory address or the 8 bits of the I/O address, 3-stated during Hold and Halt modes and during RESET.
AD_{0-7}	I/O	**MULTIPLEXED ADDRESS/DATA BUS:** Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle (T state) of a machine cycle. It then becomes the data bus during the second and third clock cycles.
ALE	O	**ADDRESS LATCH ENABLE:** It occurs during the first clock state of a machine cycle and enables the address to get latched into the on-chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. The falling edge of ALE can also be used to strobe the status information. ALE is never 3-stated.
S_0, S_1 and IO/$\overline{M}$	O	**MACHINE CYCLE STATUS:** **IO/$\overline{M}$ S_1 S_0 Status** 0 0 1 Memory write 0 1 0 Memory read 1 0 1 I/O write 1 1 0 I/O read 0 1 1 Opcode fetch 1 1 1 Interrupt Acknowledge * 0 0 Halt * X X Hold * X X Reset * = 3-state (high impedance) X = unspecified S_1 can be used as an advanced R/$\overline{W}$ status. IO/$\overline{M}$, S_0 and S_1 become valid at the beginning of a machine cycle and remain stable throughout the cycle. The falling edge of ALE may be used to latch the state of these lines.
$\overline{RD}$	O	**READ CONTROL:** A low level on $\overline{RD}$ indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer, 3-stated during Hold and Halt modes and during RESET.
$\overline{WR}$	O	**WROTE CONTROL:** A low level on $\overline{WR}$ indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of $\overline{WR}$. 3-stated during Hold and Halt modes and during RESET.
READY	I	**READY:** If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the CPU will wait an integral number of clock cycles for READY to go high before completing the read or write cycle. READY must conform to specified setup and hold times.
HOLD	I	**HOLD:** Indicates that another master is requesting the use of the address and data buses. The CPU, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ lines are 3-stated.
HLDA	O	**HOLD ACKNOWLEDGE:** Indicates that the CPU has received the HOLD request and that it will relinquish the bus in the next clock cycle. HILDA goes low after the Hold request is removed. The CPU takes the bus one half clock cycle after HLDA goes low.
INTR	I	**INTERRUPT REQUEST:** Is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of an instruction and during Hold and Halt states. If it is active, the Program Counter (PC) will be inhibited from incrementing and an $\overline{INTA}$ will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
$\overline{INTA}$	O	**INTERRUPT ACKNOWLEDGE:** Is used instead of (and has the same timing as) $\overline{RD}$ during the Instruction cycle after an INTR is accepted. It can be used to activate an 8259A Interrupt chip or some other interrupt port.
RST 5.5 RST 6.5 RST 7.5	I	**RESTART INTERRUPTS:** These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted. The priority of these interrupt is ordered as shown in Table 2. These interrupts have a higher priority than INTR. In addition, they may be individually masked out using the SIM instruction.
TRAP	I	**TRAP:** Trap interrupt is a non-maskable RESTART interrupt. It is recognized at the same time as INTR or RST 5.5–7.5. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt. (See Table 2.)
$\overline{RESET\ IN}$	I	**RESET IN:** Sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. The data and address buses and the control lines are 3-stated during RESET and because of the asynchronous nature of RESET, the processor's internal registers and flags may be altered by RESET with unpredictable results. $\overline{RESET\ IN}$ is a Schmitt-triggered input, allowing connection to an R-C network for power-on RESET delay (see Figure 3). Upon power-up, $\overline{RESET\ IN}$ must remain low for at least 10 ms after minimum V_{CC} has been reached. For proper reset operation after the power-up duration, $\overline{RESET\ IN}$ should be kept low a minimum of three clock periods. The CPU is held in the reset condition as long as $\overline{RESET\ IN}$ is applied.
RESET OUT	O	**RESET OUT:** Reset Out indicates CPU is being reset. Can be used as a system reset. The signal is synchronized to the processor clock and lasts an integral number of clock periods.
X_1, X_2	I	**X_1 and X_2:** Are connected to a crystal, LC, or RC network to drive the internal clock generator. X_1 can also be an external clock input from a logic gate. The input frequency is divided by 2 to give the processor's internal operating frequency.
CLK	O	**CLOCK:** Clock output for use as a system clock. The period of CLK is twice the X_1, X_2 input period.
SID	I	**SERIAL INPUT DATA LINE:** The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
SOD	O	**SERIAL OUTPUT DATA LINE:** The output SOD is set or reset as specified by the SIM instruction.
V_{CC}		**POWER:** +5 volt supply.
V_{SS}		**GROUND:** Reference.

Table 2. Interrupt Priority, Restart Address and Sensitivity

Name	Priority	Address Branched to(1) When Interrupt Occurs	Type Trigger
TRAP	1	24H	Rising Edge AND High Level until Sampled
RST 7.5	2	3CH	Rising Edge (Latched)
RST 6.5	3	34H	High Level until Sampled
RST 5.5	4	2CH	High Level until Sampled
INTR	5	(Note 2)	High Level until Sampled

NOTES:
1. The processor pushes the PC on the stack before branching to the indicated address.
2. The address branched to depends on the instruction provided to the CPU when the interrupt is acknowledged.

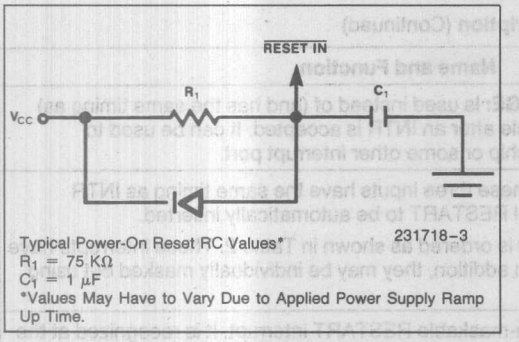


Figure 3. Power-On Reset Circuit

FUNCTIONAL DESCRIPTION

The 8085AH is a complete 8-bit parallel central processor. It is designed with N-channel, depletion load, silicon gate technology (HMOS), and requires a single +5V supply. Its basic clock speed is 3 MHz (8085AH), 5 MHz (8085AH-2), or 6 MHz (8085-AH-1), thus improving on the present 8080A's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's: The CPU (8085AH), a RAM/IO (8156H), and an EPROM/IO chip (8755A).

The 8085AH has twelve addressable 8-bit registers. Four of them can function only as two 16-bit register pairs. Six others can be used interchangeably as 8-bit registers or as 16-bit register pairs. The 8085AH register set is as follows:

Mnemonic	Register	Contents
ACC or A	Accumulator	8 Bits
PC	Program Counter	16-Bit Address
BC, DE, HL	General-Purpose Registers; data pointer (HL)	8-Bits x 6 or 16 Bits x 3
SP	Stack Pointer	16-Bit Address
Flags or F	Flag Register	5 Flags (8-Bit Space)

The 8085AH uses a multiplexed Data Bus. The address is split between the higher 8-bit Address Bus and the lower 8-bit Address/Data Bus. During the first T state (clock cycle) of a machine cycle the low order address is sent out on the Address/Data bus. These lower 8 bits may be latched externally by the Address Latch Enable signal (ALE). During the rest of the machine cycle the data bus is used for memory or I/O data.

The 8085AH provides $\overline{RD}$, $\overline{WR}$, S_0, S_1, and IO/$\overline{M}$ signals for bus control. An Interrupt Acknowledge signal ($\overline{INTA}$) is also provided. HOLD and all Interrupts are synchronized with the processor's internal clock. The 8085AH also provides Serial Input Data (SID) and Serial Output Data (SOD) lines for simple serial interface.

In addition to these features, the 8085AH has three maskable, vector interrupt pins, one nonmaskable TRAP interrupt, and a bus vectored interrupt, INTR.

INTERRUPT AND SERIAL I/O

The 8085AH has 5 interrupt inputs: INTR, RST 5.5, RST 6.5, RST 7.5, and TRAP. INTR is identical in function to the 8080A INT. Each of the three RESTART inputs, 5.5, 6.5, and 7.5, has a programmable mask. TRAP is also a RESTART interrupt but it is nonmaskable.

The three maskable interrupt cause the internal execution of RESTART (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The nonmaskable TRAP causes the internal execution of a RESTART vector independent of the state of the interrupt enable or masks. (See Table 2.)

There are two different types of inputs in the restart interrupts. RST 5.5 and RST 6.5 are *high level-sensitive* like INTR (and INT on the 8080) and are recognized with the same timing as INTR. RST 7.5 is *rising edge-sensitive.*

For RST 7.5, only a pulse is required to set an internal flip-flop which generates the internal interrupt request (a normally high level signal with a low going pulse is recommended for highest system noise immunity). The RST 7.5 request flip-flop remains set until the request is serviced. Then it is reset automatically. This flip-flop may also be reset by using the SIM instruction or by issuing a $\overline{RESET\ IN}$ to the 8085AH. The RST 7.5 internal flip-flop will be set by a pulse on the RST 7.5 pin even when the RST 7.5 interrupt is masked out.

The status of the three RST interrupt masks can only be affected by the SIM instruction and $\overline{RESET\ IN}$. (See SIM, Chapter 5 of the 8080/8085 User's Manual.)

The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP—highest priority, RST 7.5, RST 6.5, RST 5.5, INTR—lowest priority. This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST 5.5 can interrupt an RST 7.5 routine if the interrupts are re-enabled before the end of the RST 7.5 routine.

The TRAP interrupt is useful for catastrophic events such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the

highest priority. It is not affected by any flag or mask. The TRAP input is both *edge and level sensitive.* The TRAP input must go high and remain high until it is acknowledged. It will not be recognized again until it goes low, then high again. This avoids any false triggering due to noise or logic glitches. Figure 4 illustrates the TRAP interrupt request circuitry within the 8085AH. Note that the servicing of any interrupt (TRAP, RST 7.5, RST 6.5, RST 5.5, INTR) disables all future interrupts (except TRAPs) until an EI instruction is executed.

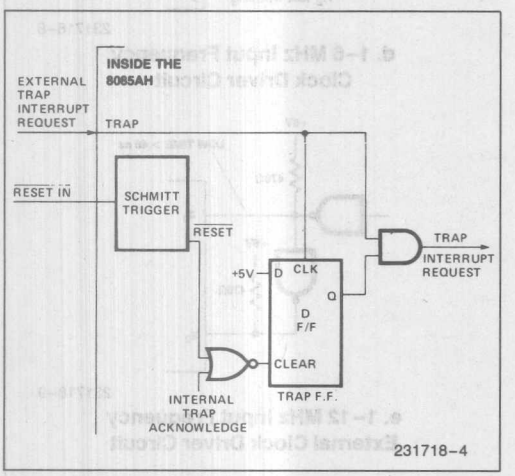


Figure 4. TRAP and RESET In Circuit

The TRAP interrupt is special in that it disables interrupts, but preserves the previous interrupt enable status. Performing the first RIM instruction following a TRAP interrupt allows you to determine whether interrupts were enabled or disabled prior to the TRAP. All subsequent RIM instructions provide current interrupt enable status. Performing a RIM instruction following INTR, or RST 5.5–7.5 will provide current Interrupt Enable status, revealing that interrupts are disabled. See the description of the RIM instruction in the 8080/8085 Family User's Manual.

The serial I/O system is also controlled by the RIM and SIM instruction. SID is read by RIM, and SIM sets the SOD data.

DRIVING THE X_1 AND X_2 INPUTS

You may drive the clock inputs of the 8085AH, 8085AH-2, or 8085AH-1 with a crystal, an LC tuned circuit, an RC network, or an external clock source. The crystal frequency must be at least 1 MHz, and must be twice the desired internal clock frequency; hence, the 8085AH is operated with a 6 MHz crystal (for 3 MHz clock), the 8085AH-2 operated with a 10 MHz crystal (for 5 MHz clock), and the 8085AH-1 can be operated with a 12 MHz crystal (for 6 MHz clock). If a crystal is used, it must have the following characteristics:

Parallel resonance at twice the clock frequency desired
C_L (load capacitance) $\leq$ 30 pF
C_S (Shunt capacitance) $\leq$ 7 pF
R_S (equivalent shunt resistance) $\leq 75\Omega$
Drive level: 10 mW
Frequency tolerance: $\pm 0.005\%$ (suggested)

Note the use of the 20 pF capacitor between X_2 and ground. This capacitor is required with crystal frequencies below 4 MHz to assure oscillator startup at the correct frequency. A parallel-resonant LC citcuit may be used as the frequency-determining network for the 8085AH, providing that its frequency tolerance of approximately $\pm 10\%$ is acceptable. The components are chosen from the formula:

$$f = \frac{1}{2\pi\sqrt{L(C_{ext} + C_{int})}}$$

To minimize variations in frequency, it is recommended that you choose a value for C_{ext} that is at least twice that of C_{int}, or 30 pF. The use of an LC circuit is not recommended for frequencies higher than approximately 5 MHz.

An RC circuit may be used as the frequency-determining network for the 8085AH if maintaining a precise clock frequency is of no importance. Variations in the on-chip timing generation can cause a wide variation in frequency when using the RC mode. Its advantage is its low component cost. The driving frequency generated by the circuit shown is approximately 3 MHz. It is not recommended that frequencies greatly higher or lower than this be attempted.

Figure 5 shows the recommended clock driver circuits. Note in d and e that pullup resistors are required to assure that the high level voltage of the input is at least 4V and maximum low level voltage of 0.8V.

For driving frequencies up to and including 6 MHz you may supply the driving signal to X_1 and leave X_2 open-circuited (Figure 5d). If the driving frequency is from 6 MHz to 12 MHz, stability of the clock generator will be improved by driving both X_1 and X_2 with a push-pull source (Figure 5e). To prevent self-oscillation of the 8085AH, be sure that X_2 is not coupled back to X_1 through the driving circuit.

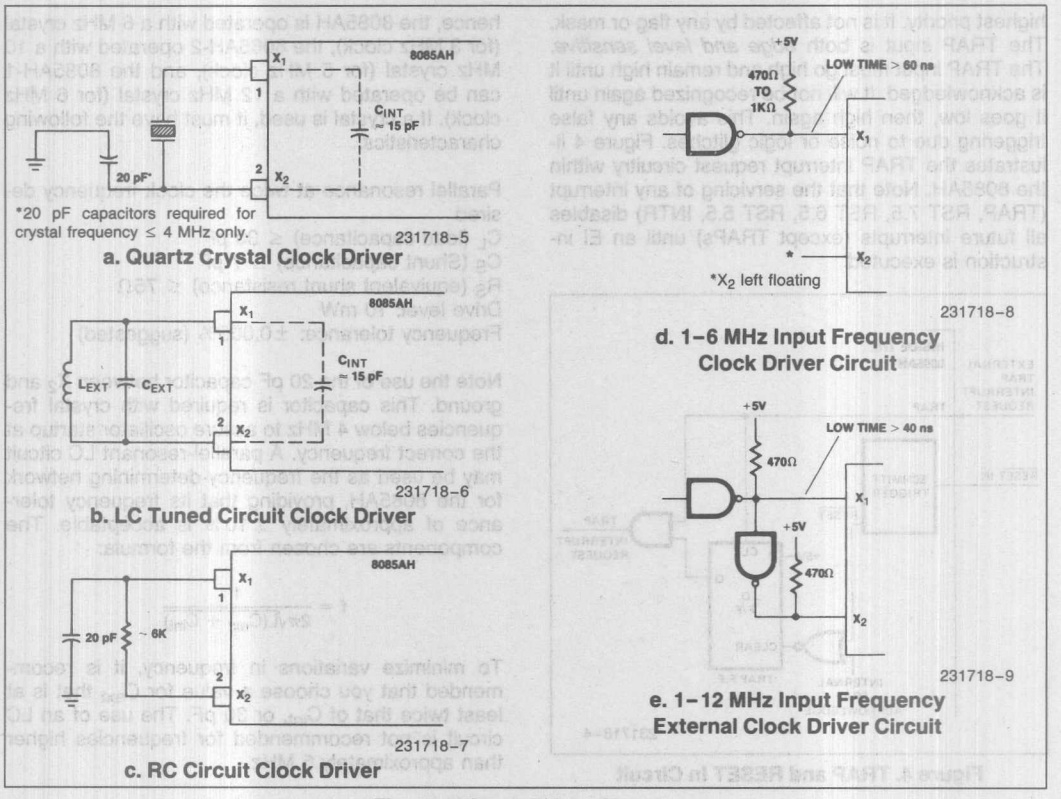


Figure 5. Clock Driver Circuits

GENERATING AN 8085AH WAIT STATE

If your system requirements are such that slow memories or peripheral devices are being used, the circuit shown in Figure 6 may be used to insert one WAIT state in each 8085AH machine cycle.

The D flip-flops should be chosen so that
- CLK is rising edge-triggered
- CLEAR is low-level active.

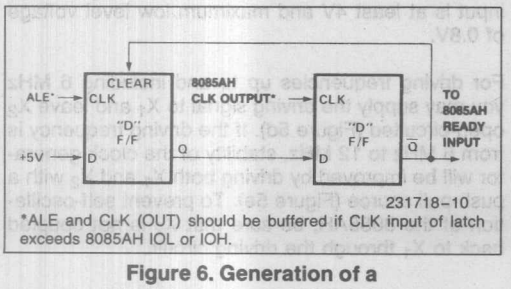


Figure 6. Generation of a Wait State for 8085AH CPU

As in the 8080, the READY line is used to extend the read and write pulse lengths so that the 8085AH can be used with slow memory. HOLD causes the CPU to relinquish the bus when it is through with it by floating the Address and Data Buses.

SYSTEM INTERFACE

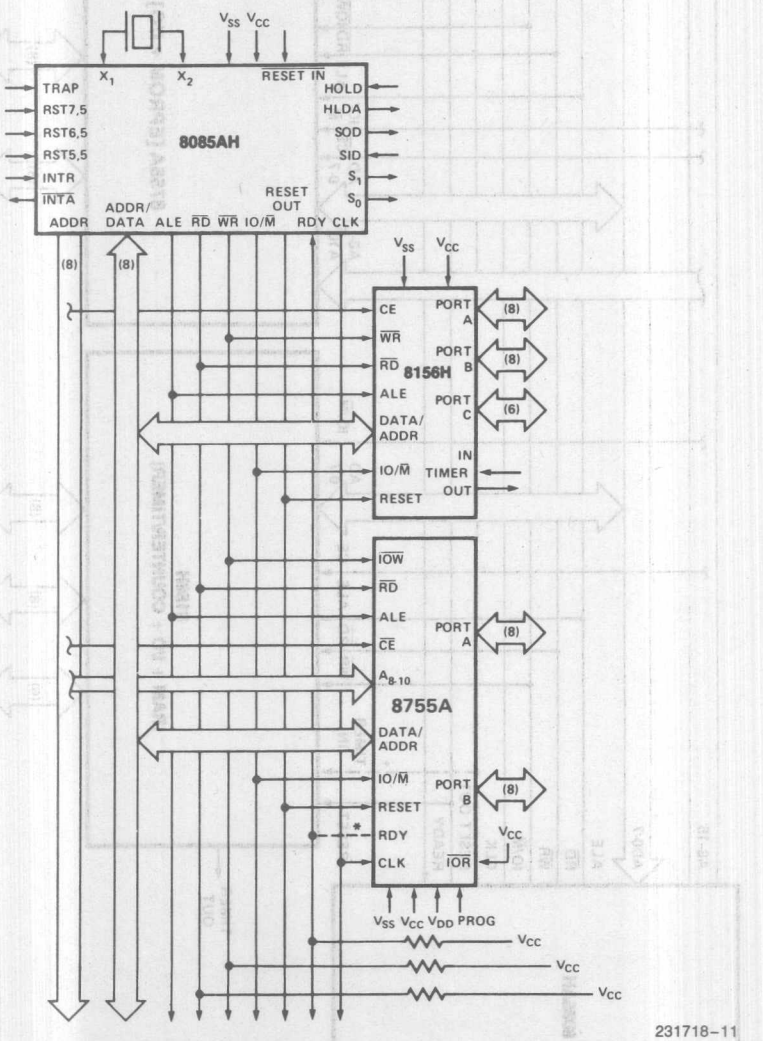
The 8085AH family includes memory components, which are directly compatible to the 8085AH CPU. For example, a system consisting of the three chips, 8085AH, 8156H and 8755A will have the following features:

- 2K Bytes EPROM
- 256 Bytes RAM
- 1 Timer/Counter
- 4 8-bit I/O Ports
- 1 6-bit I/O Port
- 4 Interrupt Levels
- Serial In/Serial Out Ports

This minimum system, using the standard I/O technique is as shown in Figure 7.

In addition to the standard I/O, the memory mapped I/O offers an efficient I/O addressing technique. With this technique, an area of memory address space is assigned for I/O address, thereby, using the memory address for I/O manipulation. Figure 8 shows the system configuration of Memory Mapped I/O using 8085AH.

The 8085AH CPU can also interface with the standard memory that does *not* have the multiplexed address/data bus. It will require a simple 8-bit latch as shown in Figure 9.



***NOTE:**
Optional Connection

Figure 7. 8085AH Minimum System (Standard I/O Technique)

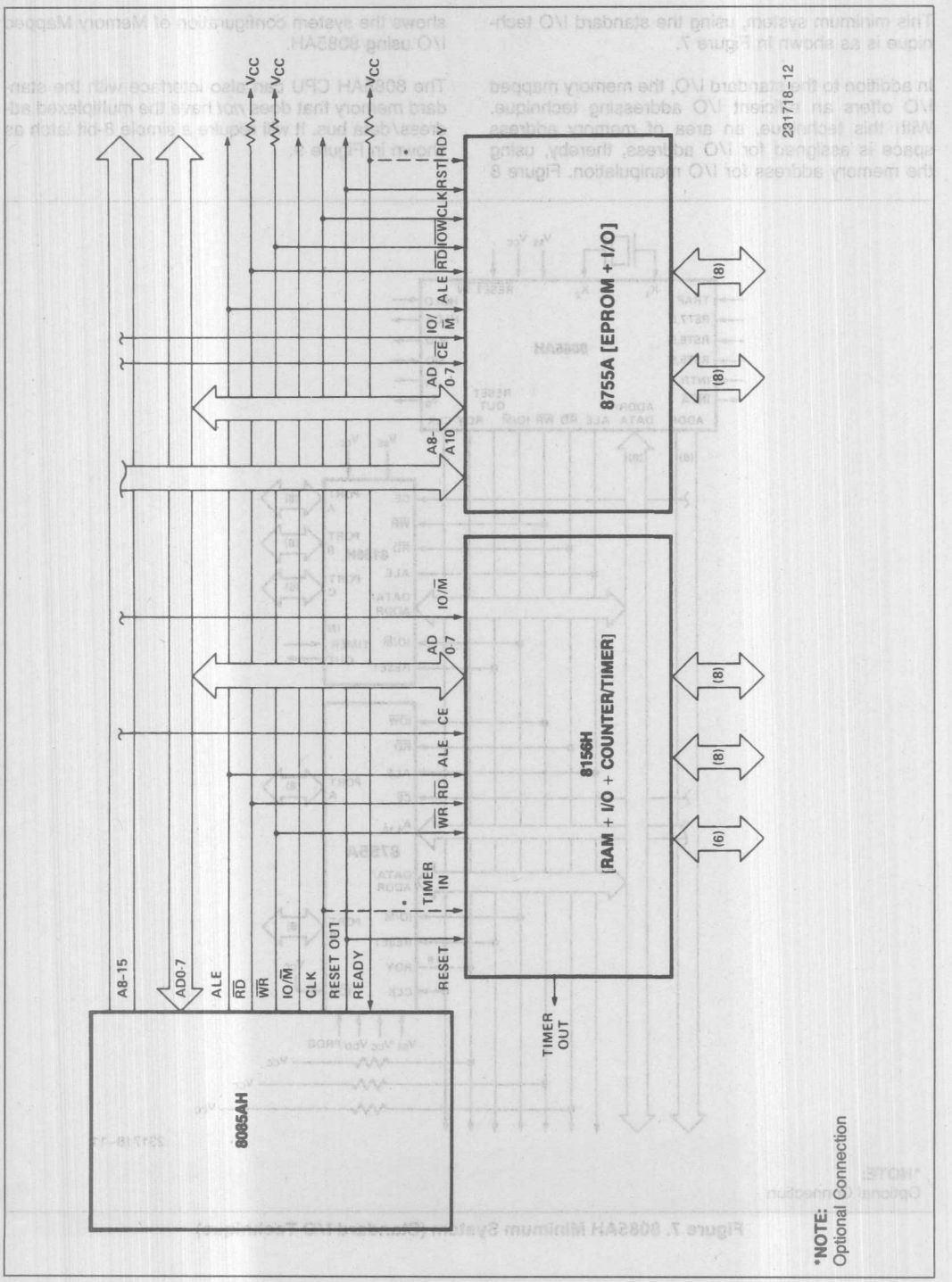


Figure 8. 8085 Minimum System (Memory Mapped I/O)

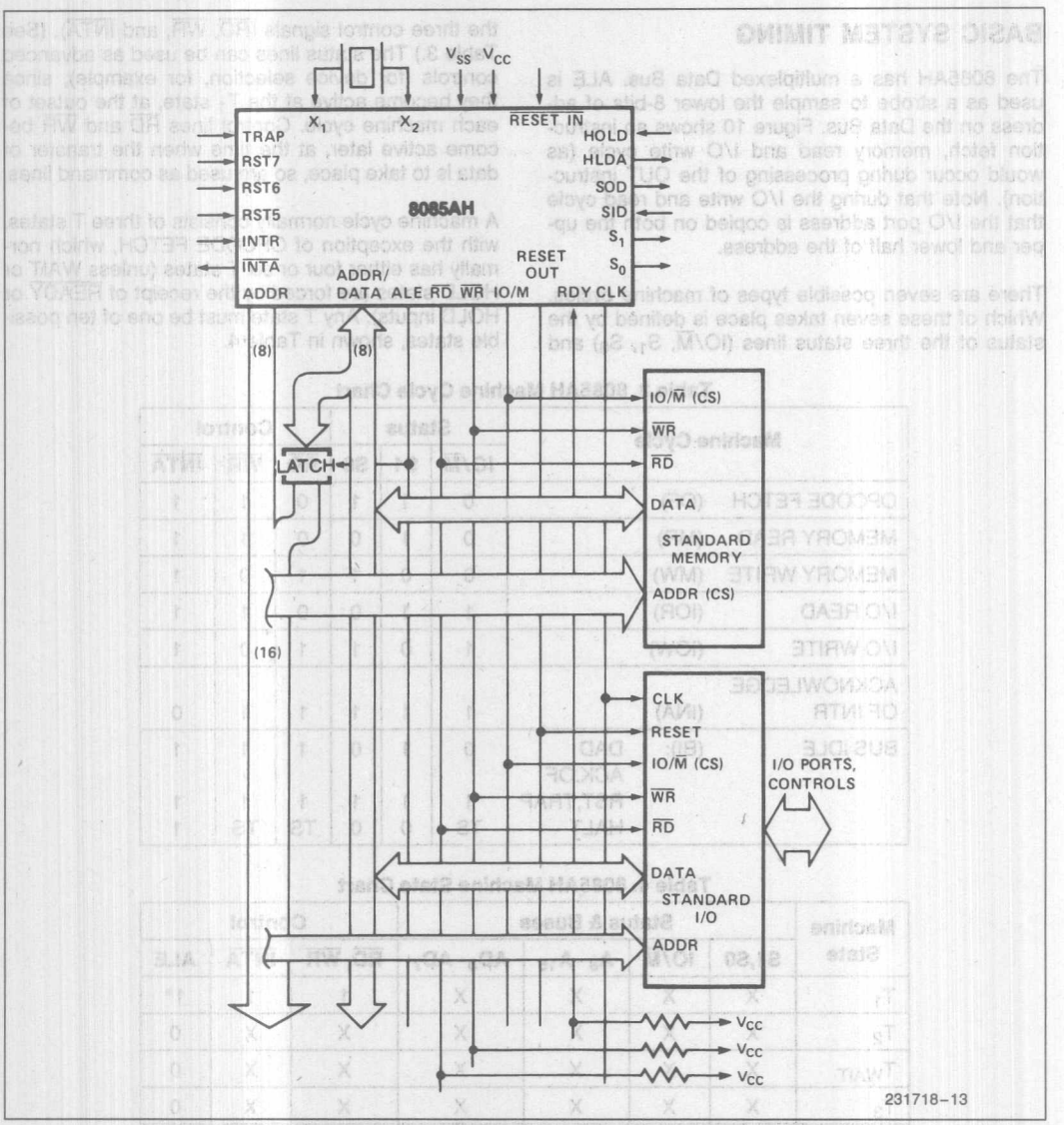


Figure 9. 8085 System (Using Standard Memories)

BASIC SYSTEM TIMING

The 8085AH has a multiplexed Data Bus. ALE is used as a strobe to sample the lower 8-bits of address on the Data Bus. Figure 10 shows an instruction fetch, memory read and I/O write cycle (as would occur during processing of the OUT instruction). Note that during the I/O write and read cycle that the I/O port address is copied on both the upper and lower half of the address.

There are seven possible types of machine cycles. Which of these seven takes place is defined by the status of the three status lines ($IO/\overline{M}$, S_1, S_0) and the three control signals ($\overline{RD}$, $\overline{WR}$, and $\overline{INTA}$). (See Table 3.) The status lines can be used as advanced controls (for device selection, for example), since they become active at the T_1 state, at the outset of each machine cycle. Control lines $\overline{RD}$ and $\overline{WR}$ become active later, at the time when the transfer of data is to take place, so are used as command lines.

A machine cycle normally consists of three T states, with the exception of OPCODE FETCH, which normally has either four or six T states (unless WAIT or HOLD states are forced by the receipt of $\overline{READY}$ or HOLD inputs). Any T state must be one of ten possible states, shown in Table 4.

Table 3. 8085AH Machine Cycle Chart

Machine Cycle			Status			Control		
			$IO/\overline{M}$	S1	S0	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
OPCODE FETCH	(OF)		0	1	1	0	1	1
MEMORY READ	(MR)		0	1	0	0	1	1
MEMORY WRITE	(MW)		0	0	1	1	0	1
I/O READ	(IOR)		1	1	0	0	1	1
I/O WRITE	(IOW)		1	0	1	1	0	1
ACKNOWLEDGE OF INTR	(INA)		1	1	1	1	1	0
BUS IDLE	(BI):	DAD	0	1	0	1	1	1
		ACK.OF RST,TRAP	1	1	1	1	1	1
		HALT	TS	0	0	TS	TS	1

Table 4. 8085AH Machine State Chart

Machine State	Status & Buses				Control		
	S1,S0	$IO/\overline{M}$	A_8–A_{15}	AD_0–AD_7	$\overline{RD}$, $\overline{WR}$	$\overline{INTA}$	ALE
T_1	X	X	X	X	1	1	1*
T_2	X	X	X	X	X	X	0
T_{WAIT}	X	X	X	X	X	X	0
T_3	X	X	X	X	X	X	0
T_4	1	0†	X	TS	1	1	0
T_5	1	0†	X	TS	1	1	0
T_6	1	0†	X	TS	1	1	0
T_{RESET}	X	TS	TS	TS	TS	1	0
T_{HALT}	0	TS	TS	TS	TS	1	0
T_{HOLD}	X	TS	TS	TS	TS	1	0

0 = Logic "0" TS = High Impedance
1 = Logic "1" X = Unspecified
*ALE not generated during 2nd and 3rd machine cycles of DAD instruction.
†$IO/\overline{M}$ = 1 during T_4–T_6 of INA machine cycle.

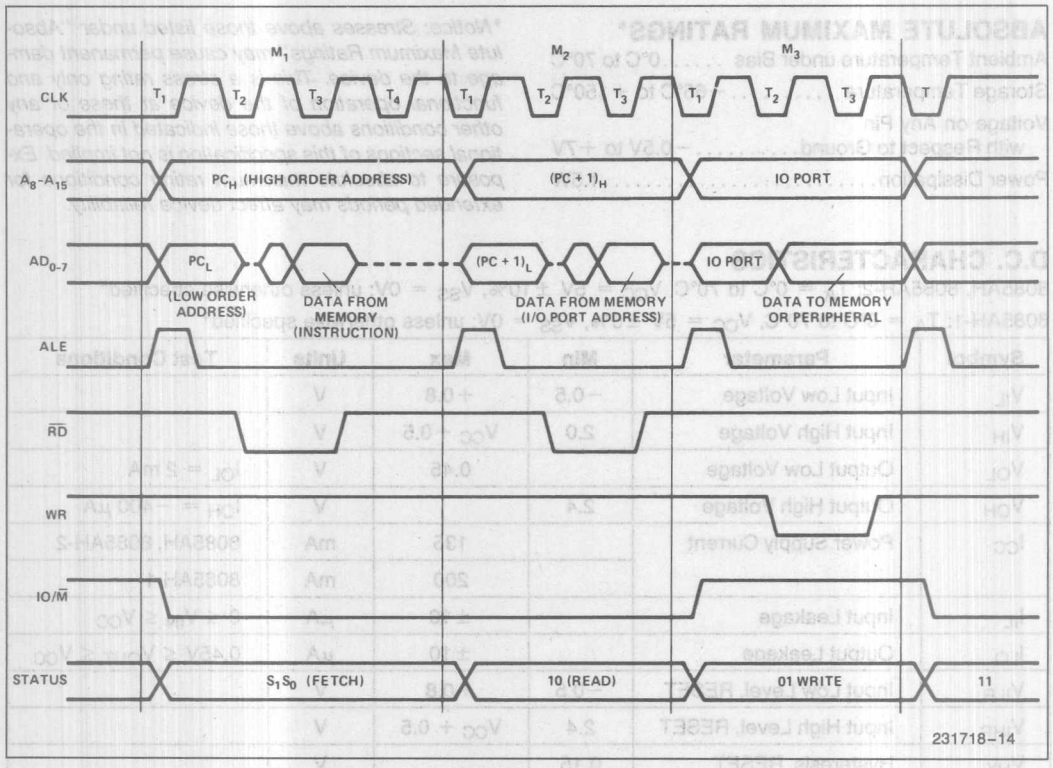


Figure 10. 8085AH Basic System Timing

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin
with Respect to Ground.....−0.5V to +7V
Power Dissipation.....1.5W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

8085AH, 8085AH-2: T_A = 0°C to 70°C, V_{CC} = 5V ±10%, V_{SS} = 0V; unless otherwise specified*
8085AH-1: T_A = 0°C to 70°C, V_{CC} = 5V ±5%, V_{SS} = 0V; unless otherwise specified*

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	+0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} +0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
I_{CC}	Power Supply Current		135	mA	8085AH, 8085AH-2
			200	mA	8085AH-1
I_{IL}	Input Leakage		±10	μA	$0 \le V_{IN} \le V_{CC}$
I_{LO}	Output Leakage		±10	μA	$0.45V \le V_{OUT} \le V_{CC}$
V_{ILR}	Input Low Level, RESET	−0.5	+0.8	V	
V_{IHR}	Input High Level, RESET	2.4	V_{CC} + 0.5	V	
V_{HY}	Hysteresis, RESET	0.15		V	

A.C. CHARACTERISTICS

8085AH, 8085AH-2: T_A = 0°C to 70°C, V_{CC} = 5V ±10%, V_{SS} = 0V*
8085AH-1: T_A = 0°C to 70°C, V_{CC} = 5V ±5%, V_{SS} = 0V

Symbol	Parameter	8085AH (2)		8085AH-2 (2)		8085AH-1 (2)		Units
		Min	Max	Min	Max	Min	Max	
t_{CYC}	CLK Cycle Period	320	2000	200	2000	167	2000	ns
t_1	CLK Low Time (Standard CLK Loading)	80		40		20		ns
t_2	CLK High Time (Standard CLK Loading)	120		70		50		ns
t_r, t_f	CLK Rise and Fall Time		30		30		30	ns
t_{XKR}	X_1 Rising to CLK Rising	20	120	20	100	20	100	ns
t_{XKF}	X_1 Rising to CLK Falling	20	150	20	110	20	110	ns
t_{AC}	A_{8-15} Valid to Leading Edge of Control (1)	270		115		70		ns
t_{ACL}	A_{0-7} Valid to Leading Edge of Control	240		115		60		ns
t_{AD}	A_{0-15} Valid to Valid Data In		575		350		225	ns
t_{AFR}	Address Float after Leading Edge of $\overline{\text{READ}}$ ($\overline{\text{INTA}}$)		0		0		0	ns
t_{AL}	A_{8-15} Valid before Trailing Edge of ALE (1)	115		50		25		ns

***NOTE:**
For Extended Temperature EXPRESS use M8085AH Electricals Parameters.

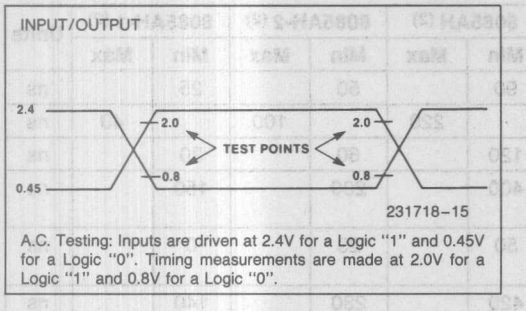
A.C. CHARACTERISTICS (Continued)

Symbol	Parameter	8085AH (2)		8085AH-2 (2)		8085AH-1 (2)		Units
		Min	Max	Min	Max	Min	Max	
t_{ALL}	A_{0-7} Valid before Trailing Edge of ALE	90		50		25		ns
t_{ARY}	READY Valid from Address Valid		220		100		40	ns
t_{CA}	Address (A_{8-15}) Valid after Control	120		60		30		ns
t_{CC}	Width of Control Low ($\overline{RD}$, $\overline{WR}$, $\overline{INTA}$) Edge of ALE	400		230		150		ns
t_{CL}	Trailing Edge of Control to Leading Edge of ALE	50		25		0		ns
t_{DW}	Data Valid to Trialing Edge of $\overline{WRITE}$	420		230		140		ns
t_{HABE}	HLDA to Bus Enable		210		150		150	ns
t_{HABF}	Bus Float after HLDA		210		150		150	ns
t_{HACK}	HLDA Valid to Trailing Edge of CLK	110		40		0		ns
t_{HDH}	HOLD Hold Time	0		0		0		ns
t_{HDS}	HOLD Setup Time to Trailing Edge of CLK	170		120		120		ns
t_{INH}	INTR Hold Time	0		0		0		ns
t_{INS}	INTR, RST, and TRAP Setup Time to Falling Edge of CLK	160		150		150		ns
t_{LA}	Address Hold Time after ALE	100		50		20		ns
t_{LC}	Trailing Edge of ALE to Leading Edge of Control	130		60		25		ns
t_{LCK}	ALE Low During CLK High	100		50		15		ns
t_{LDR}	ALE to Valid Data during Read		460		270		175	ns
t_{LDW}	ALE to Valid Data during Write		200		140		110	ns
t_{LL}	ALE Width	140		80		50		ns
t_{LRY}	ALE to READY Stable		110		30		10	ns
t_{RAE}	Trailing Edge of $\overline{READ}$ to Re-Enabling of Address	150		90		50		ns
t_{RD}	$\overline{READ}$ (or $\overline{INTA}$) to Valid Data		300		150		75	ns
t_{RV}	Control Trailing Edge to Leading Edge of Next Control	400		220		160		ns
t_{RDH}	Data Hold Time after $\overline{READ}$ $\overline{INTA}$	0		0		0		ns
t_{RYH}	READY Hold Time	0		0		5		ns
t_{RYS}	READY Setup Time to Leading Edge of CLK	110		100		100		ns
t_{WD}	Data Valid after Trailing Edge of $\overline{WRITE}$	100		60		30		ns
t_{WDL}	LEADING Edge of $\overline{WRITE}$ to Data Valid		40		20		30	ns

NOTES:

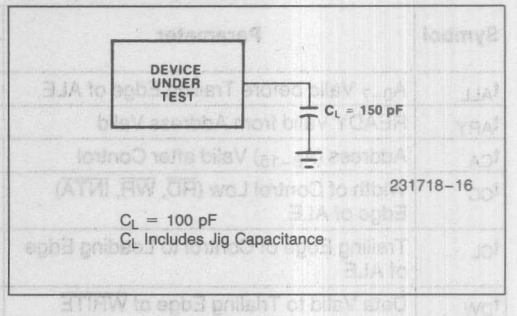
1. A_8–A_{15} address Specs apply IO/$\overline{M}$, S_0, and S_1 except A_8–A_{15} are undefined during T_4–T_6 of OF cycle whereas IO/$\overline{M}$, S_0, and S_1 are stable.
2. *Test Conditions:* t_{CYC} = 320 ns (8085AH)/200 ns (8085AH-2);/167 ns (8085AH-1); C_L = 150 pF.
3. For all output timing where C ≠ 150 pF use the following correction factors:
   25 pF ≤ C_L < 150 pF: −0.10 ns/pF
   150 pF < C_L ≤ 300 pF: +0.30 ns/pF
4. Output timings are measured with purely capacitive load.
5. To calculate timing specifications at other values of t_{CYC} use Table 5.

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT



C_L = 100 pF
C_L Includes Jig Capacitance

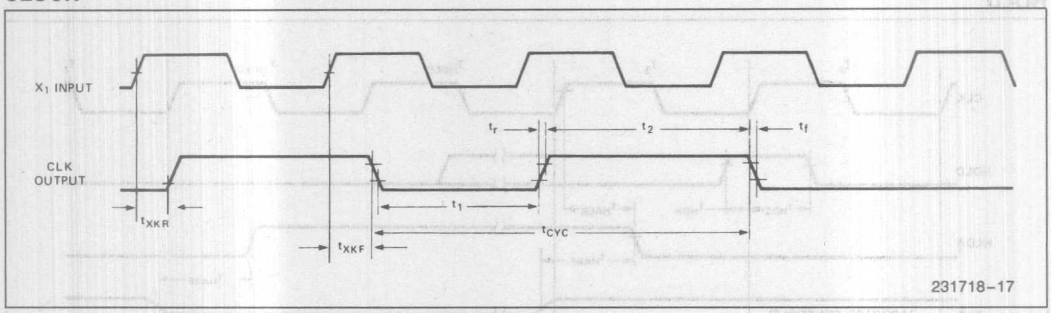
Table 5. Bus Timing Specification as a T_{CYC} Dependent

Symbol	8085AH	8085AH-2	8085AH-1	
t_{AL}	(1/2)T − 45	(1/2)T − 50	(1/2)T − 58	Minimum
t_{LA}	(1/2)T − 60	(1/2)T − 50	(1/2)T − 63	Minimum
t_{LL}	(1/2)T − 20	(1/2)T − 20	(1/2)T − 33	Minimum
t_{LCK}	(1/2)T − 60	(1/2)T − 50	(1/2)T − 68	Minimum
t_{LC}	(1/2)T − 30	(1/2)T − 40	(1/2)T − 58	Minimum
t_{AD}	(5/2 + N)T − 225	(5/2 + N)T − 150	(5/2 + N)T − 192	Maximum
t_{RD}	(3/2 + N)T − 180	(3/2 + N)T − 150	(3/2 + N)T − 175	Maximum
t_{RAE}	(1/2)T − 10	(1/2)T − 10	(1/2)T − 33	Minimum
t_{CA}	(1/2)T − 40	(1/2)T − 40	(1/2)T − 53	Minimum
t_{DW}	(3/2 + N)T − 60	(3/2 + N)T − 70	(3/2 + N)T − 110	Minimum
t_{WD}	(1/2)T − 60	(1/2)T − 40	(1/2)T − 53	Minimum
t_{CC}	(3/2 + N)T − 80	(3/2 + N)T − 70	(3/2 + N)T − 100	Minimum
t_{CL}	(1/2)T − 110	(1/2)T − 75	(1/2)T − 83	Minimum
t_{ARY}	(3/2)T − 260	(3/2)T − 200	(3/2)T − 210	Maximum
t_{HACK}	(1/2)T − 50	(1/2)T − 60	(1/2)T − 83	Minimum
t_{HABF}	(1/2)T + 50	(1/2)T + 50	(1/2)T + 67	Maximum
t_{HABE}	(1/2)T + 50	(1/2)T + 50	(1/2)T + 67	Maximum
t_{AC}	(2/2)T − 50	(2/2)T − 85	(2/2)T − 97	Minimum
t_1	(1/2)T − 80	(1/2)T − 60	(1/2)T − 63	Minimum
t_2	(1/2)T − 40	(1/2)T − 30	(1/2)T − 33	Minimum
t_{RV}	(3/2)T − 80	(3/2)T − 80	(3/2)T − 90	Minimum
t_{LDR}	(4/2 + N)T − 180	(4/2)T − 130	(4/2)T − 159	Maximum

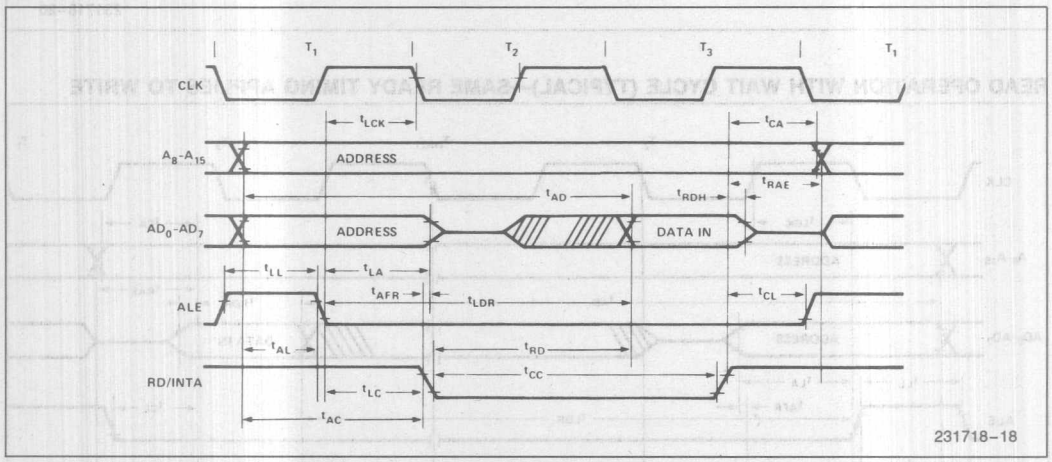
NOTE:
N is equal to the total WAIT states. T = t_{CYC}.

WAVEFORMS

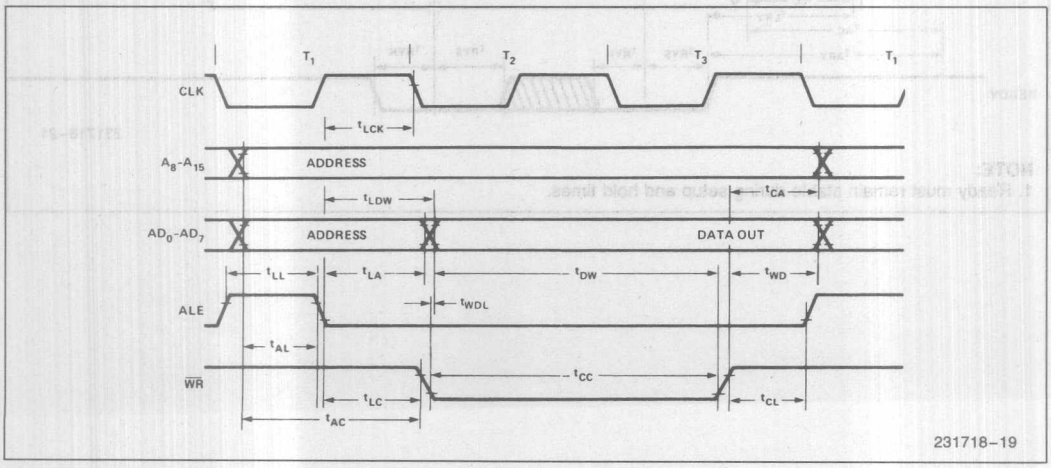
CLOCK



READ

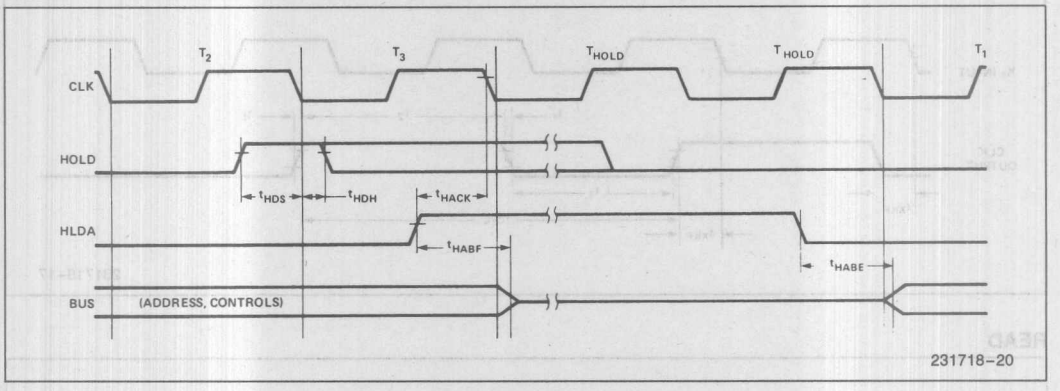


WRITE

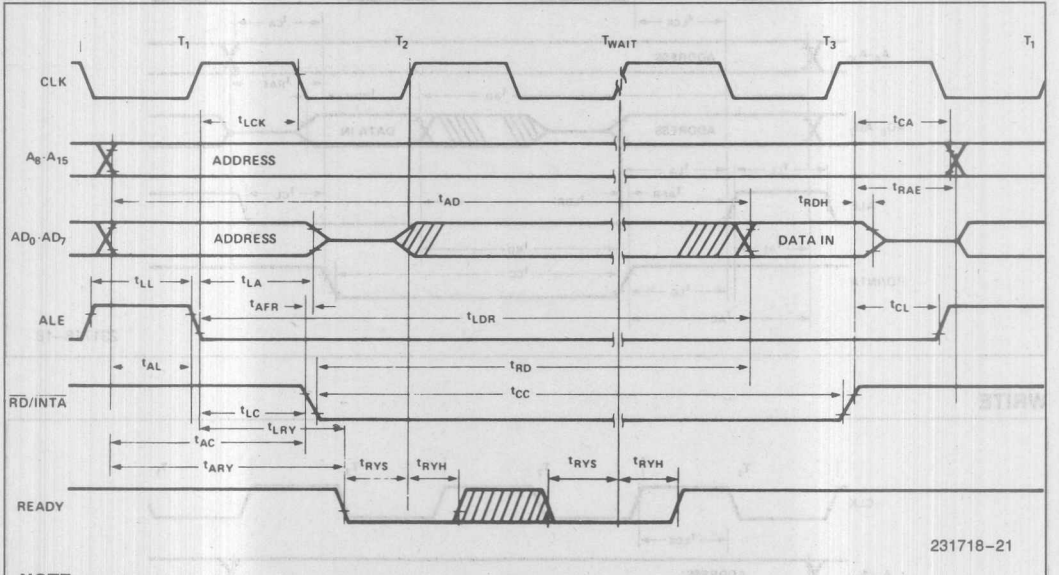


WAVEFORMS (Continued)

HOLD



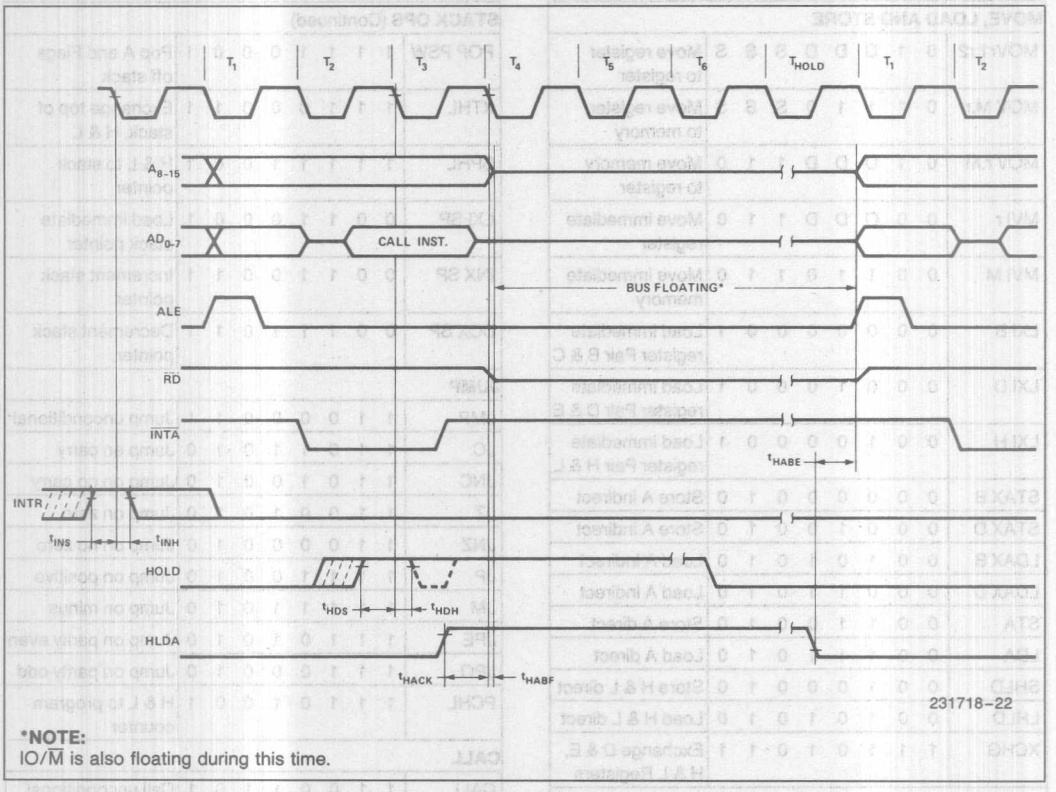
READ OPERATION WITH WAIT CYCLE (TYPICAL)—SAME READY TIMING APPLIES TO WRITE



NOTE:
1. Ready must remain stable during setup and hold times.

WAVEFORMS (Continued)

INTERRUPT AND HOLD



***NOTE:**
IO/$\overline{M}$ is also floating during this time.

Table 6. Instruction Set Summary

Mnemonic	Instruction Code D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Operations Description
MOVE, LOAD AND STORE									
MOVr1 r2	0	1	D	D	D	S	S	S	Move register to register
MOV M.r	0	1	1	1	0	S	S	S	Move register to memory
MOV r.M	0	1	D	D	D	1	1	0	Move memory to register
MVI r	0	0	D	D	D	1	1	0	Move immediate register
MVI M	0	0	1	1	0	1	1	0	Move immediate memory
LXI B	0	0	0	0	0	0	0	1	Load immediate register Pair B & C
LXI D	0	0	0	1	0	0	0	1	Load immediate register Pair D & E
LXI H	0	0	1	0	0	0	0	1	Load immediate register Pair H & L
STAX B	0	0	0	0	0	0	1	0	Store A indirect
STAX D	0	0	0	1	0	0	1	0	Store A indirect
LDAX B	0	0	1	0	1	0	1	0	Load A indirect
LDAX D	0	0	0	1	1	0	1	0	Load A indirect
STA	0	0	1	1	0	0	1	0	Store A direct
LDA	0	0	1	1	1	0	1	0	Load A direct
SHLD	0	0	1	0	0	0	1	0	Store H & L direct
LHLD	0	0	1	0	1	0	1	0	Load H & L direct
XCHG	1	1	1	0	1	0	1	1	Exchange D & E, H & L Registers
STACK OPS									
PUSH B	1	1	0	0	0	1	0	1	Push register Pair B & C on stack
PUSH D	1	1	0	1	0	1	0	1	Push register Pair D & E on stack
PUSH H	1	1	1	0	0	1	0	1	Push register Pair H & L on stack
PUSH PSW	1	1	1	1	0	1	0	1	Push A and Flags on stack
POP B	1	1	0	0	0	0	0	1	Pop register Pair B & C off stack
POP D	1	1	0	1	0	0	0	1	Pop register Pair D & E off stack
POP H	1	1	1	0	0	0	0	1	Pop register Pair H & L off stack

Mnemonic	Instruction Code D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Operations Description
STACK OPS (Continued)									
POP PSW	1	1	1	1	0	0	0	1	Pop A and Flags off stack
XTHL	1	1	1	0	0	0	1	1	Exchange top of stack, H & L
SPHL	1	1	1	1	1	0	0	1	H & L to stack pointer
LXI SP	0	0	1	1	0	0	0	1	Load immediate stack pointer
INX SP	0	0	1	1	0	0	1	1	Increment stack pointer
DCX SP	0	0	1	1	1	0	1	1	Decrement stack pointer
JUMP									
JMP	1	1	0	0	0	0	1	1	Jump unconditional
JC	1	1	0	1	1	0	1	0	Jump on carry
JNC	1	1	0	1	0	0	1	0	Jump on no carry
JZ	1	1	0	0	1	0	1	0	Jump on zero
JNZ	1	1	0	0	0	0	1	0	Jump on no zero
JP	1	1	1	1	0	0	1	0	Jump on positive
JM	1	1	1	1	1	0	1	0	Jump on minus
JPE	1	1	1	0	1	0	1	0	Jump on parity even
JPO	1	1	1	0	0	0	1	0	Jump on parity odd
PCHL	1	1	1	0	1	0	0	1	H & L to program counter
CALL									
CALL	1	1	0	0	1	1	0	1	Call unconditional
CC	1	1	0	1	1	1	0	0	Call on carry
CNC	1	1	0	1	0	1	0	0	Call on no carry
CZ	1	1	0	0	1	1	0	0	Call on zero
CNZ	1	1	0	0	0	1	0	0	Call on no zero
CP	1	1	1	1	0	1	0	0	Call on positive
CM	1	1	1	1	1	1	0	0	Call on minus
CPE	1	1	1	0	1	1	0	0	Call on parity even
CPO	1	1	1	0	0	1	0	0	Call on parity odd
RETURN									
RET	1	1	0	0	1	0	0	1	Return
RC	1	1	0	1	1	0	0	0	Return on carry
RNC	1	1	0	1	0	0	0	0	Return on no carry
RZ	1	1	0	0	1	0	0	0	Return on zero

Table 6. Instruction Set Summary (Continued)

Mnemonic	Instruction Code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Operations Description
RETURN (Continued)		
RNZ	1 1 0 0 0 0 0 0	Return on no zero
RP	1 1 1 1 0 0 0 0	Return on positive
RM	1 1 1 1 1 0 0 0	Return on minus
RPE	1 1 1 0 1 0 0 0	Return on parity even
RPO	1 1 1 0 0 0 0 0	Return on parity odd
RESTART		
RST	1 1 A A A 1 1 1	Restart
INPUT/OUTPUT		
IN	1 1 0 1 1 0 1 1	Input
OUT	1 1 0 1 0 0 1 1	Output
INCREMENT AND DECREMENT		
INR r	0 0 D D D 1 0 0	Increment register
DCR r	0 0 D D D 1 0 1	Decrement register
INR M	0 0 1 1 0 1 0 0	Increment memory
DCR M	0 0 1 1 0 1 0 1	Decrement memory
INX B	0 0 0 0 0 0 1 1	Increment B & C registers
INX D	0 0 0 1 0 0 1 1	Increment D & E registers
INX H	0 0 1 0 0 0 1 1	Increment H & L registers
DCX B	0 0 0 0 1 0 1 1	Decrement B & C
DCX D	0 0 0 1 1 0 1 1	Decrement D & E
DCX H	0 0 1 0 1 0 1 1	Decrement H & L
ADD		
ADD r	1 0 0 0 0 S S S	Add register to A
ADC r	1 0 0 0 1 S S S	Add register to A with carry
ADD M	1 0 C 0 0 1 1 0	Add memory to A
ADC M	1 0 0 0 1 1 1 0	Add memory to A with carry
ADI	1 1 0 0 0 1 1 0	Add immediate to A
ACI	1 1 0 0 1 1 1 0	Add immediate to A with carry
DAD B	0 0 0 0 1 0 0 1	Add B & C to H & L

Mnemonic	Instruction Code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Operations Description
ADD (Continued)		
DAD D	0 0 0 1 1 0 0 1	Add D & E to H & L
DAD H	0 0 1 0 1 0 0 1	Add H & L to H & L
DAD SP	0 0 1 1 1 0 0 1	Add stack pointer to H & L
SUBTRACT		
SUB r	1 0 0 1 0 S S S	Subtract register from A
SBB r	1 0 0 1 1 S S S	Subtract register from A with borrow
SUB M	1 0 0 1 0 1 1 0	Subtract memory from A
SBB M	1 0 0 1 1 1 1 0	Subtract memory from A with borrow
SUI	1 1 0 1 0 1 1 0	Subtract immediate from A
SBI	1 1 0 1 1 1 1 0	Subtract immediate from A with borrow
LOGICAL		
ANA r	1 0 1 0 0 S S S	And register with A
XRA r	1 0 1 0 1 S S S	Exclusive OR register with A
ORA r	1 0 1 1 0 S S S	OR register with A
CMP r	1 0 1 1 1 S S S	Compare register with A
ANA M	1 0 1 0 0 1 1 0	And memory with A
XRA M	1 0 1 0 1 1 1 0	Exclusive OR memory with A
ORA M	1 0 1 1 0 1 1 0	OR memory with A
CMP M	1 0 1 1 1 1 1 0	Compare memory with A
ANI	1 1 1 0 0 1 1 0	And immediate with A
XRI	1 1 1 0 1 1 1 0	Exclusive OR immediate with A
ORI	1 1 1 1 0 1 1 0	OR immediate with A
CPI	1 1 1 1 1 1 1 0	Compare immediate with A

Table 6. Instruction Set Summary (Continued)

Mnemonic	Instruction Code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Operations Description
ROTATE		
RLC	0 0 0 0 0 1 1 1	Rotate A left
RRC	0 0 0 0 1 1 1 1	Rotate A right
RAL	0 0 0 1 0 1 1 1	Rotate A left through carry
RAR	0 0 0 1 1 1 1 1	Rotate A right through carry
SPECIALS		
CMA	0 0 1 0 1 1 1 1	Complement A
STC	0 0 1 1 0 1 1 1	Set carry
CMC	0 0 1 1 1 1 1 1	Complement carry
DAA	0 0 1 0 0 1 1 1	Decimal adjust A

Mnemonic	Instruction Code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Operations Description
CONTROL		
EI	1 1 1 1 1 0 1 1	Enable Interrupts
DI	1 1 1 1 0 0 1 1	Disable Interrupt
NOP	0 0 0 0 0 0 0 0	No-operation
HLT	0 1 1 1 0 1 1 0	Halt
NEW 8085AH INSTRUCTIONS		
RIM	0 0 1 0 0 0 0 0	Read Interrupt Mask
SIM	0 0 1 1 0 0 0 0	Set Interrupt Mask

NOTES:
1. DDS or SSS: B 000, C 001, D 010, E011, H 100, L101, Memory 110, A 111.
2. Two possible cycle times (6/12) indicate instruction cycles dependent on condition flags.

*All mnemonics copyrighted ©Intel Corporation 1976.

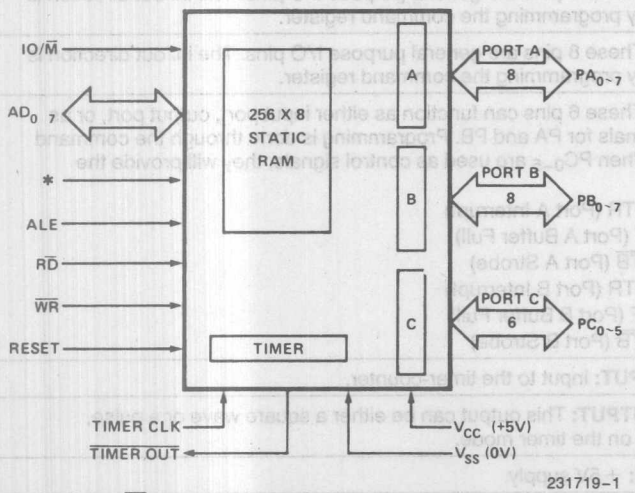
8155H/8156H/8155H-2/8156H-2
2048-BIT STATIC HMOS RAM WITH I/O PORTS AND TIMER

- **Single +5V Power Supply with 10% Voltage Margins**
- **30% Lower Power Consumption than the 8155 and 8156**
- **256 Word x 8 Bits**
- **Completely Static Operation**
- **Internal Address Latch**
- **2 Programmable 8-Bit I/O Ports**
- **1 Programmable 6-Bit I/O Port**
- **Programmable 14-Bit Binary Counter/ Timer**
- **Compatible with 8085AH and 8088 CPU**
- **Multiplexed Address and Data Bus**
- **Available in EXPRESS**
  - **— Standard Temperature Range**
  - **— Extended Temperature Range**

The Intel® 8155H and 8156H are RAM and I/O chips implemented in N-Channel, depletion load, silicon gate technology (HMOS), to be used in the 8085AH and 8088 microprocessor systems. The RAM portion is designed with 2048 static cells organized as 256 x 8. They have a maximum access time of 400 ns to permit use with no wait states in 8085AH CPU. The 8155H-2 and 8156H-2 have maximum access times of 330 ns for use with the 8085H-2 and the 5 MHz 8088 CPU.

The I/O portion consists of three general purpose I/O ports. One of the three ports can be programmed to be status pins, thus allowing the other two ports to operate in handshake mode.

A 14-bit programmable counter/timer is also included on chip to provide either a square wave or terminal count pulse for the CPU system depending on timer mode.



*8155H/8155H-2 = $\overline{CE}$, 8156H/8156H-2 = CE

Figure 1. Block Diagram

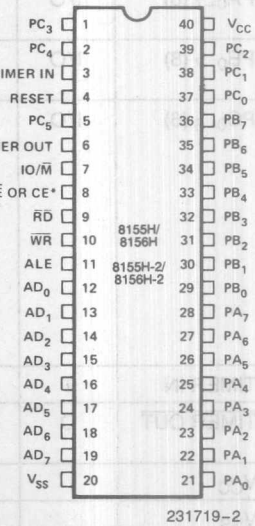


Figure 2. Pin Configuration

December 1986
Order Number: 231719-001

Symbol	Type	Name and Function
RESET	I	**RESET:** Pulse provided by the 8085AH to initialize the system (connect to 8085AH RESET OUT). Input high on this line resets the chip and initializes the three I/O ports to input mode. The width of RESET pulse should typically be two 8085AH clock cycle times.
AD_{0-7}	I/O	**ADDRESS/DATA:** 3-state Address/Data lines that interface with the CPU lower 8-bit Address/Data Bus. The 8-bit address is latched into the address latch inside the 8155H/56H on the falling edge of ALE. The address can be either for the memory section or the I/O section depending on the IO/$\overline{M}$ input. The 8-bit data is either written into the chip or read from the chip, depending on the $\overline{WR}$ or $\overline{RD}$ input signal.
CE or $\overline{CE}$	I	**CHIP ENABLE:** On the 8155H, this pin is $\overline{CE}$ and is ACTIVE LOW. On the 8156H, this pin is CE and is ACTIVE HIGH.
$\overline{RD}$	I	**READ CONTROL:** Input low on this line with the Chip Enable active enables and AD_{0-7} buffers. If IO/$\overline{M}$ pin is low, the RAM content will be read out to the AD bus. Otherwise the content of the selected I/O port or command/status registers will be read to the AD bus.
$\overline{WR}$	I	**WRITE CONTROL:** Input low on this line with the Chip Enable active causes the data on the Address/Data bus to be written to the RAM or I/O ports and command/status register, depending on IO/$\overline{M}$.
ALE	I	**ADDRESS LATCH ENABLE:** This control signal latches both the address on the AD_{0-7} lines and the state of the Chip Enable and IO/$\overline{M}$ into the chip at the falling edge of ALE.
IO/$\overline{M}$	I	**I/O MEMORY:** Selects memory if low and I/O and command/status registers if high.
PA_{0-7} (8)	I/O	**PORT A:** These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register.
PB_{0-7} (8)	I/O	**PORT B:** These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register.
PC_{0-5} (6)	I/O	**PORT C:** These 6 pins can function as either input port, output port, or as control signals for PA and PB. Programming is done through the command register. When PC_{0-5} are used as control signals, they will provide the following: PC_0—A INTR (Port A Interrupt) PC_1—ABF (Port A Buffer Full) PC_2—$\overline{A\ STB}$ (Port A Strobe) PC_3—B INTR (Port B Interrupt) PC_4—B BF (Port B Buffer Full) PC_5—$\overline{B\ STB}$ (Port B Strobe)
TIMER IN	I	**TIMER INPUT:** Input to the timer-counter.
$\overline{TIMER\ OUT}$	O	**TIMER OUTPUT:** This output can be either a square wave or a pulse, depending on the timer mode.
V_{CC}		**VOLTAGE:** +5V supply.
V_{SS}		**GROUND:** Ground reference.

FUNCTIONAL DESCRIPTION

The 8155H/8156H contains the following:

- 2K Bit Static RAM organized as 256 x 8
- Two 8-bit I/O ports (PA & PB) and one 6-bit I/O port (PC)
- 14-bit timer-counter

The IO/$\overline{M}$ (IO/Memory Select) pin selects either the five registers (Command, Status, PA_{0-7}, PB_{0-7}, PC_{0-5}) or the memory (RAM) portion.

The 8-bit address on the Address/Data lines, Chip Enable input CE or $\overline{CE}$, and IO/$\overline{M}$ are all latched on-chip at the falling edge of ALE.

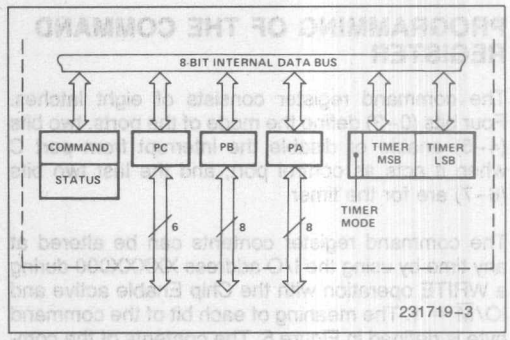
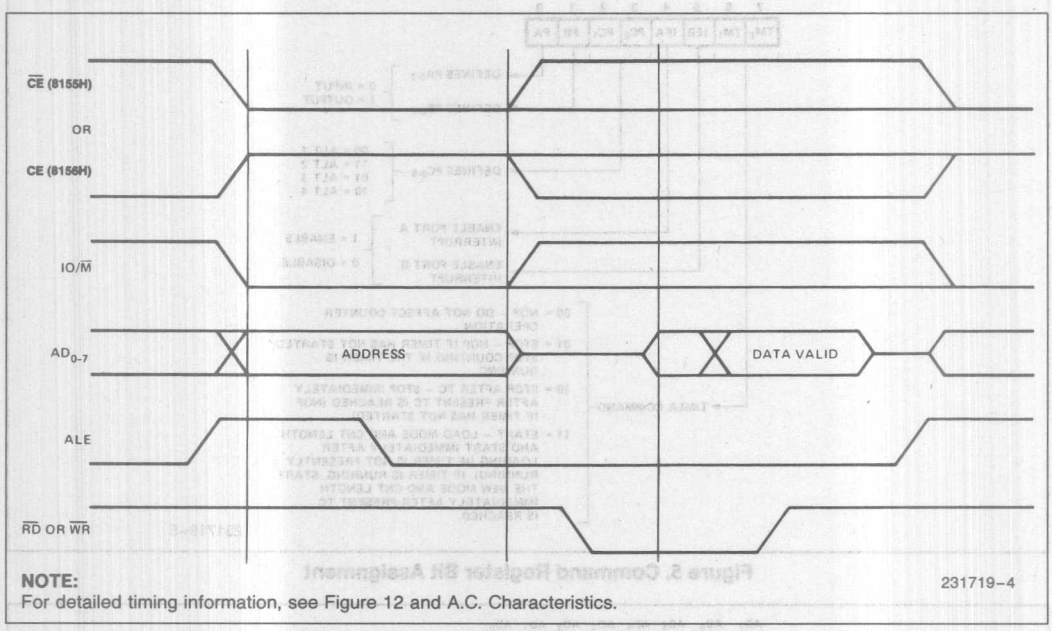


Figure 3. 8155H/8156H Internal Registers



NOTE:
For detailed timing information, see Figure 12 and A.C. Characteristics.

Figure 4. 8155H/8156H On-Board Memory Read/Write Cycle

PROGRAMMING OF THE COMMAND REGISTER

The command register consists of eight latches. Four bits (0–3) define the mode of the ports, two bits (4–5) enable or disable the interrupt from port C when it acts as control port, and the last two bits (6–7) are for the timer.

The command register contents can be altered at any time by using the I/O address XXXXX000 during a WRITE operation with the Chip Enable active and IO/$\overline{M}$ = 1. The meaning of each bit of the command byte is defined in Figure 5. The contents of the command register may never be read.

READING THE STATUS REGISTER

The status register consists of seven latches, one for each bit; six (0–5) for the status of the ports and one (6) for the status of the timer.

The status of the timer and the I/O section can be polled by reading the Status Register (Address XXXXX000). Status word format is shown in Figure 6. Note that you may never write to the status register since the command register shares the same I/O address and the command register is selected when a write to that address is issued.

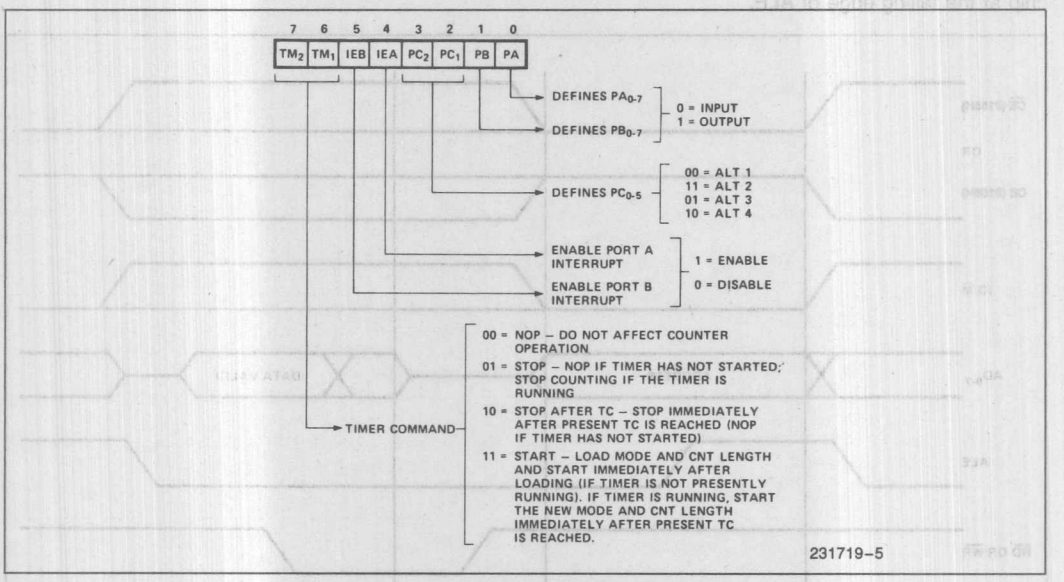


Figure 5. Command Register Bit Assignment

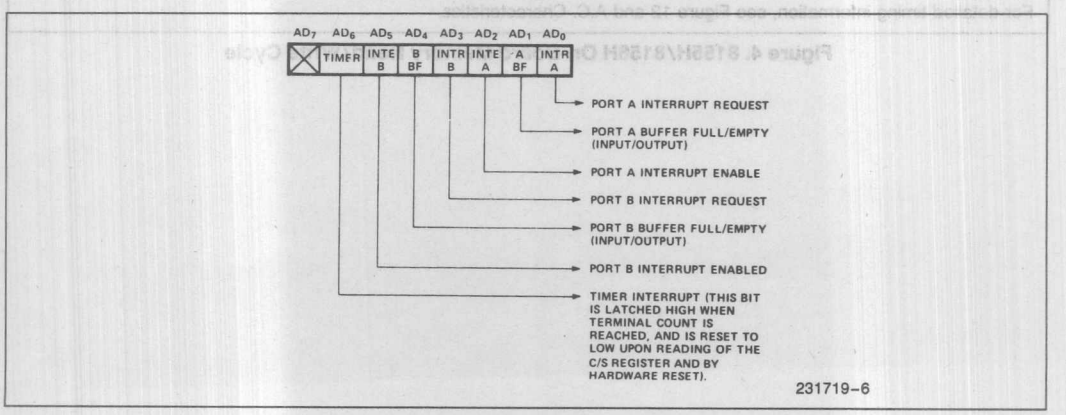


Figure 6. Status Register Bit Assignment

INPUT/OUTPUT SECTION

The I/O section of the 8155H/8156H consists of five registers: (see Figure 7.)

- **Command/Status Register (C/S)**—Both registers are assigned the address XXXXX000. The C/S address serves the dual purpose.
  When the C/S registers are selected during WRITE operation, a command is written into the command register. The contents of this register are *not* accessible through the pins.
  When the C/S (XXXXX000) is selected during a READ operation, the status information of the I/O ports and the timer becomes available on the AD_{0-7} lines.
- **PA Register**—This register can be programmed to be either input or output ports depending on the status of the contents of the C/S Register. Also depending on the command, this port can operate in either the basic mode or the strobed mode (see timing diagram). The I/O pins assigned in relation to this register are PA_{0-7}. The address of this register is XXXXX001.
- **PB Register**—This register functions the same as PA Register. The I/O pins assigned are PB_{0-7}. The address of this register is XXXXX010.
- **PC Register**—This register has the address XXXXX011 and contains only 6 bits. The 6 bits can be programmed to be either input ports, output ports or as control signals for PA and PB by properly programming the AD_2 and AD_3 bits of the C/S register.
  When PC_{0-5} is used as a control port, 3 bits are assigned for Port A and 3 for Port B. The first bit is an interrupt that the 8155H sends out. The second is an output signal indicating whether the buffer is full or empty, and the third is an input pin to accept a strobe for the strobed input mode. (See Table 2.)

When the 'C' port is programmed to either ALT3 or ALT4, the control signals for PA and PB are initialized as follows:

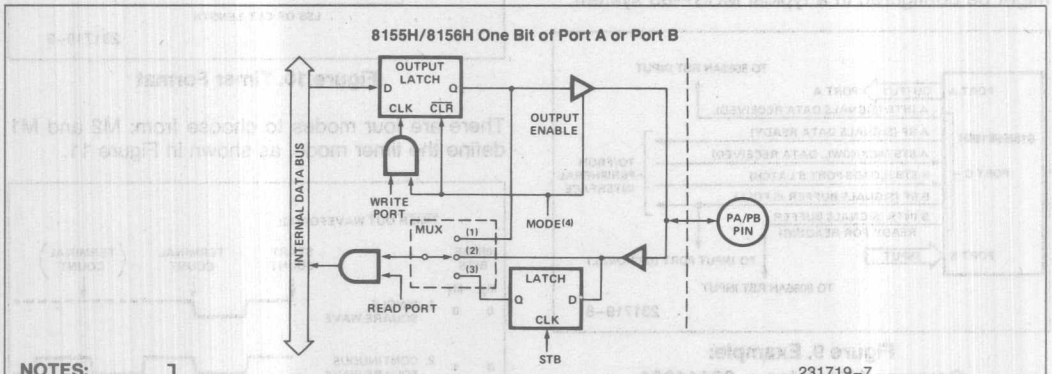
Control	Input Mode	Output Mode
BF	Low	Low
INTR	Low	High
STB	Input Control	Input Control

I/O Address†								Selection
A7	A6	A5	A4	A3	A2	A1	A0	
X	X	X	X	X	0	0	0	Interval Command/Status Register
X	X	X	X	X	0	0	1	General Purpose I/O Port A
X	X	X	X	X	0	1	0	General Purpose I/O Port B
X	X	X	X	X	0	1	1	Port C—General Purpose I/O or Control
X	X	X	X	X	1	0	0	Low-Order 8 bits of Timer Count
X	X	X	X	X	1	0	1	High 6 bits of Timer Count and 2 bits of Timer Mode

X: Don't Care.
†: I/O Address must be qualified by CE = 1 (8156H) or $\overline{CE}$ = 0 (8155H) and IO/$\overline{M}$ = 1 in order to select the appropriate register.

Figure 7. I/O Port and Timer Addressing Scheme

Figure 8 shows how I/O PORTS A and B are structured within the 8155H and 8156H:



NOTES:
(1) Output Mode
(2) Simple Input
(3) Strobed Input
Multiplexer Control

(4) = 1 for Output Mode
= 0 for Input Mode

READ Port = (IO/$\overline{M}$ = 1) • ($\overline{RD}$ = 0) • (CE Active) • (Port Address Selected)
WRITE Port = (IO/$\overline{M}$ = 1) • ($\overline{WR}$ = 0) • (CE Active) • (Port Address Selected)

Figure 8. 8155H/8156H Port Functions

Table 2. Port Control Assignment

Pin	ALT 1	ALT 2	ALT 3	ALT 4
PC0	Input Port	Output Port	A INTR (Port A Interrupt)	A INTR (Port A Interrupt)
PC1	Input Port	Output Port	A BF (Port A Buffer Full)	A BF (Port A Buffer Full)
PC2	Input Port	Output Port	A $\overline{STB}$ (Port A Strobe)	A $\overline{STB}$ (Port A Strobe)
PC3	Input Port	Output Port	Output Port	B INTR (Port B Interrupt)
PC4	Input Port	Output Port	Output Port	B BF (Port B Buffer Full)
PC5	Input Port	Output Port	Output Port	B $\overline{STB}$ (Port B Strobe)

Note in the diagram that when the I/O ports are programmed to be output ports, the contents of the output ports can still be read by a READ operation when appropriately addressed.

The outputs of the 8155H/8156H are "glitch-free" meaning that you can write a "1" to a bit position that was previously "1" and the level at the output pin will not change.

Note also that the output latch is cleared when the port enters the input mode. The output latch cannot be loaded by writing to the port if the port is in the input mode. The result is that each time a port mode is changed from input to output, the output pin will go low. When the 8155H/56H is RESET, the output latches are all cleared and all 3 ports enter the input mode.

When in the ALT 1 or ALT 2 modes, the bits of PORT C are structured like the diagram above in the simple input or output mode, respectively.

Reading from an input port with nothing connected to the pins will provide unpredictable results.

Figure 9 shows how the 8155H/8156H I/O ports might be configured in a typical MCS®-85 system.

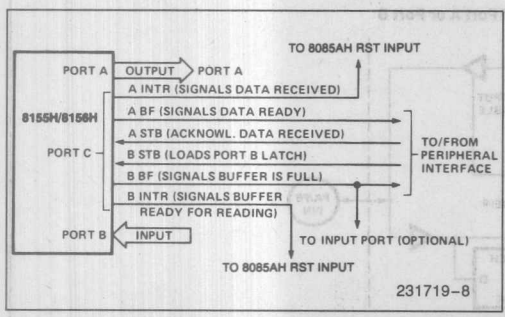


Figure 9. Example: Command Register = 00111001

TIMER SECTION

The time is a 14-bit down-counter that counts the TIMER IN pulses and provides either a square wave or pulse when terminal count (TC) is reached.

The timer has the I/O address XXXXX100 for the low order byte of the register and the I/O address XXXXX101 for the high order byte of the register. (See Figure 7.)

To program the timer, the COUNT LENGTH REG is loaded first, one byte at a time, by selecting the timer addresses. Bits 0–13 of the high order count register will specify the length of the next count and bits 14–15 of the high order register will specify the timer output mode (see Figure 10). The value loaded into the count length register can have any value from 2H through 3FFFH in Bits 0–13.

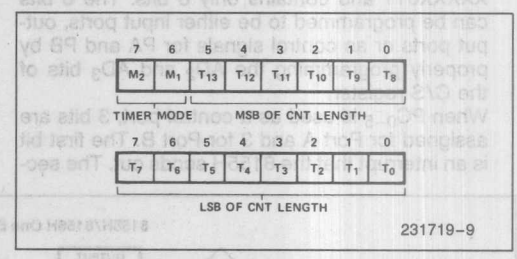


Figure 10. Timer Format

There are four modes to choose from: M2 and M1 define the timer mode, as shown in Figure 11.

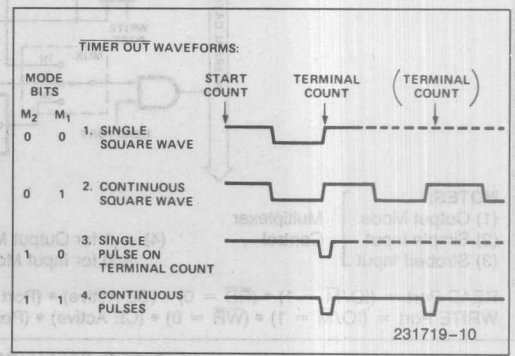


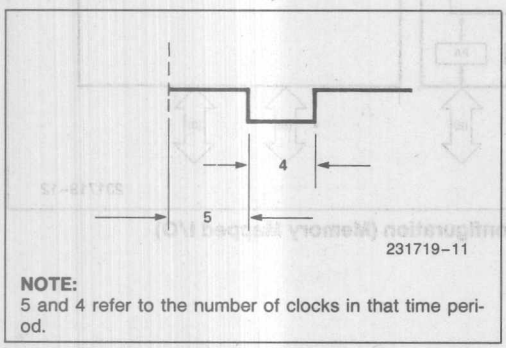
Figure 11. Timer Modes

Bits 6–7 (TM_2 and TM_1) of command register contents are used to start and stop the counter. There are four commands to choose from:

TM_2	TM_1	
0	0	NOP—Do not affect counter operation.
0	1	STOP—NOP if timer has not started; stop counting if the timer is running.
1	0	STOP AFTER TC—Stop immediately after present TC is reached (NOP if timer has not started)
1	1	START—Load mode and CNT length and start immediately after loading (if timer is not presently running). If timer is running, start the new mode and CNT length immediately after present TC is reached.

Note that while the counter is counting, you may load a new count and mode into the count length registers. Before the new count and mode will be used by the counter, you **must** issue a START command to the counter. This applies even though you may only want to change the count and use the previous mode.

In case of an odd-numbered count, the first half-cycle of the squarewave output, which is high, is one count longer than the second (low) half-cycle, as shown in Figure 12.



NOTE:
5 and 4 refer to the number of clocks in that time period.

Figure 12. Asymmetrical Square-Wave Output Resulting from Count of 9

The counter in the 8155H is not initialized to any particular mode or count when hardware RESET occurs, but RESET does *stop* the counting. Therefore, counting cannot begin following RESET until a START command is issued via the C/S register.

Please note that the timer circuit on the 8155H/8156H chip is designed to be a square-wave timer, not an event counter. To achieve this, it counts down by twos twice in completing one cycle. Thus, its registers do not contain values directly representing the number of TIMER IN pulses received. You cannot load an initial value of 1 into the count register and cause the timer to operate, as its terminal count value is 10 (binary) or 2 (decimal). (For the detection of single pulses, it is suggested that one of the hardware interrupt pins on the 8085AH be used.) After the timer has started counting down, the values residing in the count registers can be used to calculate the actual number of TIMER IN pulses required to complete the timer cycle if desired. To obtain the remaining count, perform the following operations in order:

1. Stop the count
2. Read in the 16-bit value from the count length registers
3. Reset the upper two mode bits
4. Reset the carry and rotate right one position all 16 bits through carry
5. If carry is set, add ½ of the full original count (½ full count—1 if full count is odd).

NOTE:
If you started with an odd count and you read the count length register before the third count pulse occurs, you will not be able to discern whether one or two counts has occurred. Regardless of this, the 8155H/56H always counts out the right number of pulses in generating the $\overline{\text{TIMER OUT}}$ waveforms.

8085AH MINIMUM SYSTEM CONFIGURATION

Figure 13a shows a minimum system using three chips, containing:

- 256 Bytes RAM
- 2K Bytes EPROM
- 38 I/O Pins
- 1 Interval Timer
- 4 Interrupt Levels

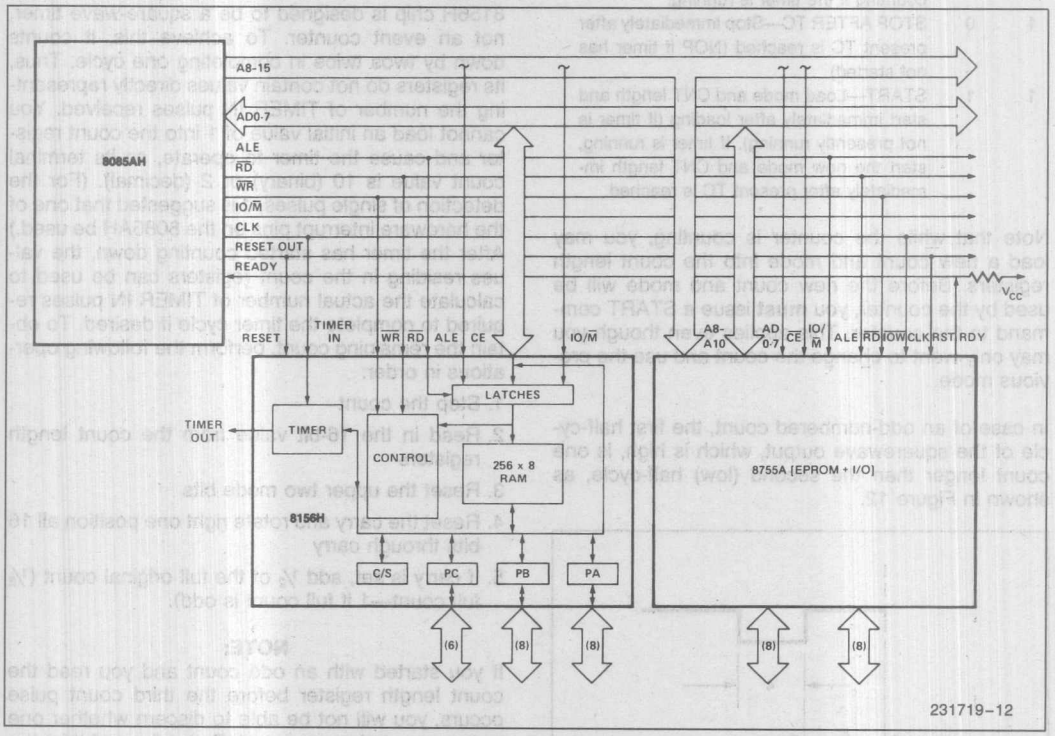


Figure 13a. 8085AH Minimum System Configuration (Memory Mapped I/O)

8088 FIVE CHIP SYSTEM

Figure 13b shows a five chip system containing:

- 1.25K Bytes RAM
- 2K Bytes EPROM
- 38 I/O Pins
- 1 Interval Timer
- 2 Interrupt Levels

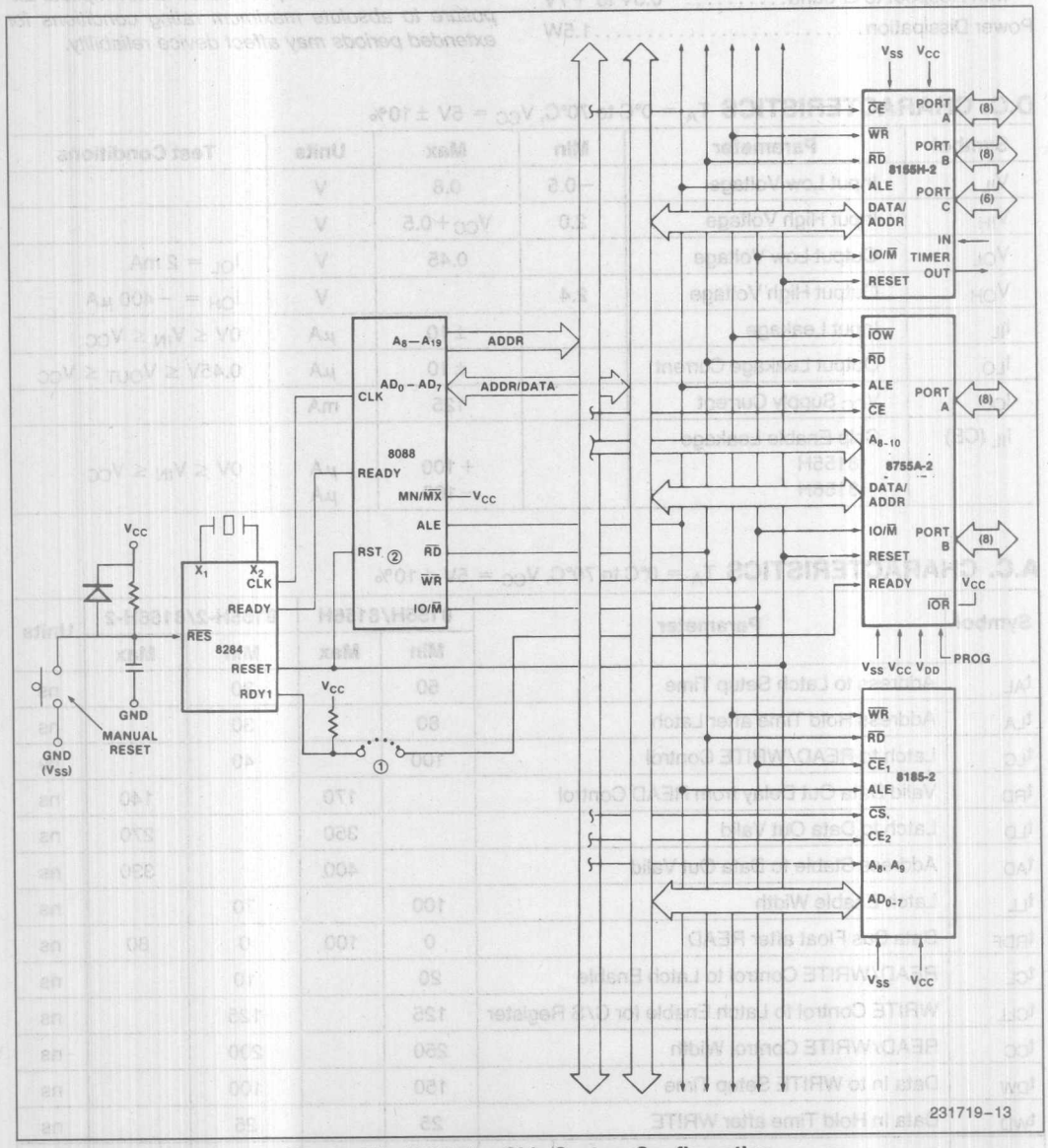


Figure 13b. 8088 Five Chip System Configuration

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin
with Respect to Ground . . . . . −0.5V to +7V
Power Dissipation 1.5W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
I_{IL}	Input Leakage		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current		±10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	V_{CC} Supply Current		125	mA	
I_{IL} (CE)	Chip Enable Leakage 8155H 8156H		 +100 −100	 μA μA	0V ≤ V_{IN} ≤ V_{CC}

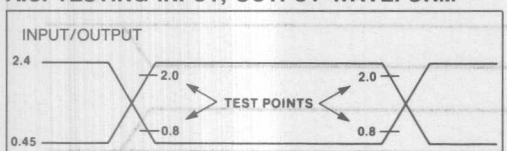
A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±10%

Symbol	Parameter	8155H/8156H		8155H-2/8156H-2		Units
		Min	Max	Min	Max	
t_{AL}	Address to Latch Setup Time	50		30		ns
t_{LA}	Address Hold Time after Latch	80		30		ns
t_{LC}	Latch to READ/WRITE Control	100		40		ns
t_{RD}	Valid Data Out Delay from READ Control		170		140	ns
t_{LD}	Latch to Data Out Valid		350		270	ns
t_{AD}	Address Stable to Data Out Valid		400		330	ns
t_{LL}	Latch Enable Width	100		70		ns
t_{RDF}	Data Bus Float after READ	0	100	0	80	ns
t_{CL}	READ/WRITE Control to Latch Enable	20		10		ns
t_{CLL}	WRITE Control to Latch Enable for C/S Register	125		125		ns
t_{CC}	READ/WRITE Control Width	250		200		ns
t_{DW}	Data In to WRITE Setup Time	150		100		ns
t_{WD}	Data In Hold Time after WRITE	25		25		ns
t_{RV}	Recovery Time between Controls	300		200		ns
t_{WP}	*WRITE to Port Output*		400		300	ns

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±10% (Continued)

Symbol	Parameter	8155H/8156H		8155H-2/8156H-2		Units
		Min	Max	Min	Max	
t_{PR}	Port Input Setup Time	70		50		ns
t_{RP}	Port Input Hold Time	50		10		ns
t_{SBF}	Strobe to Buffer Full		400		300	ns
t_{SS}	Strobe Width	200		150		ns
t_{RBE}	READ to Buffer Empty		400		300	ns
t_{SI}	Strobe to INTR On		400		300	ns
t_{RDI}	READ to INTR Off		400		300	ns
t_{PSS}	Port Setup Time to Strobe	50		0		ns
t_{PHS}	Port Hold Time After Strobe	120		100		ns
t_{SBE}	Strobe to Buffer Empty		400		300	ns
t_{WBF}	WRITE to Buffer Full		400		300	ns
t_{WI}	WRITE to INTR Off		400		300	ns
t_{TL}	TIMER-IN to $\overline{\text{TIMER-OUT}}$ Low		400		300	ns
t_{TH}	TIMER-IN to $\overline{\text{TIMER-OUT}}$ High		400		300	ns
t_{RDE}	Data Bus Enable from READ Control	10		10		ns
t_1	TIMER-IN Low Time	80		40		ns
t_2	TIMER-IN High Time	120		70		ns
t_{WT}	WRITE to TIMER-IN (for writes which start counting)	360		200		ns

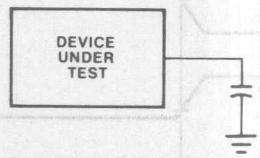
A.C. TESTING INPUT, OUTPUT WAVEFORM



231719-14

A.C. testing: inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".

A.C. TESTING LOAD CIRCUIT

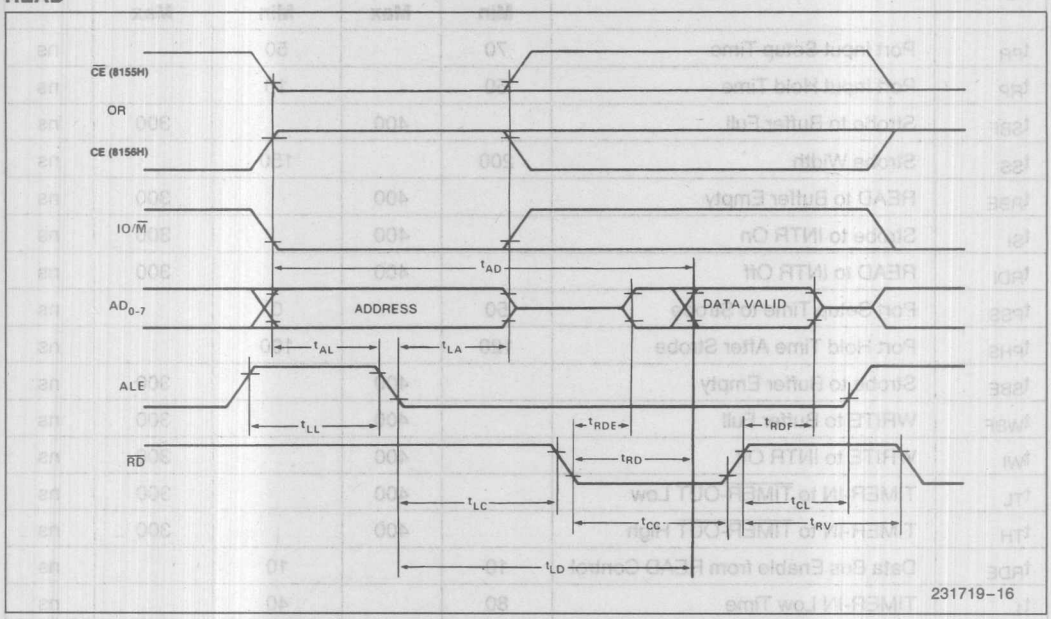


231719-15

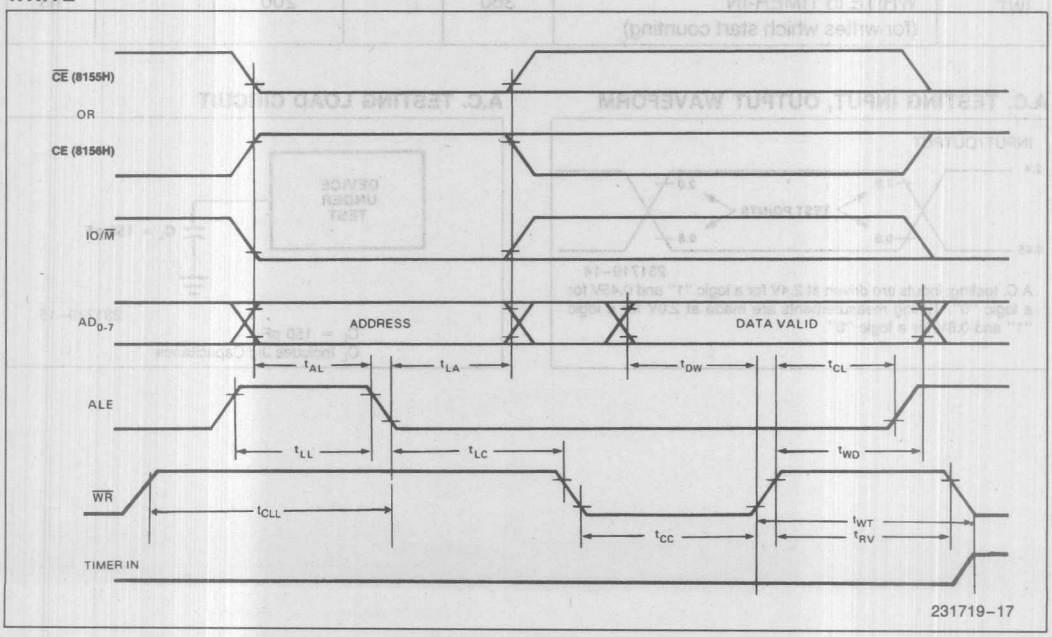
C_L = 150 pF
C_L Includes Jig Capacitance

WAVEFORMS

READ

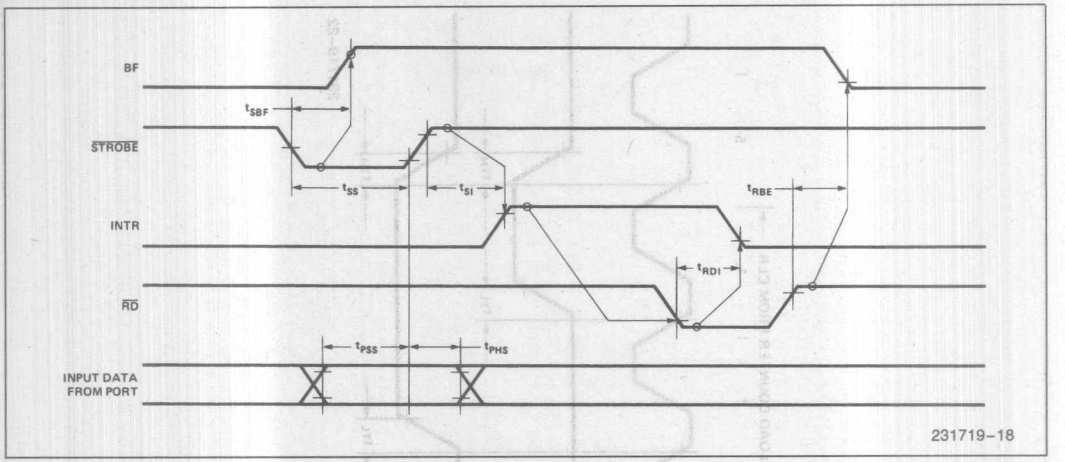


WRITE

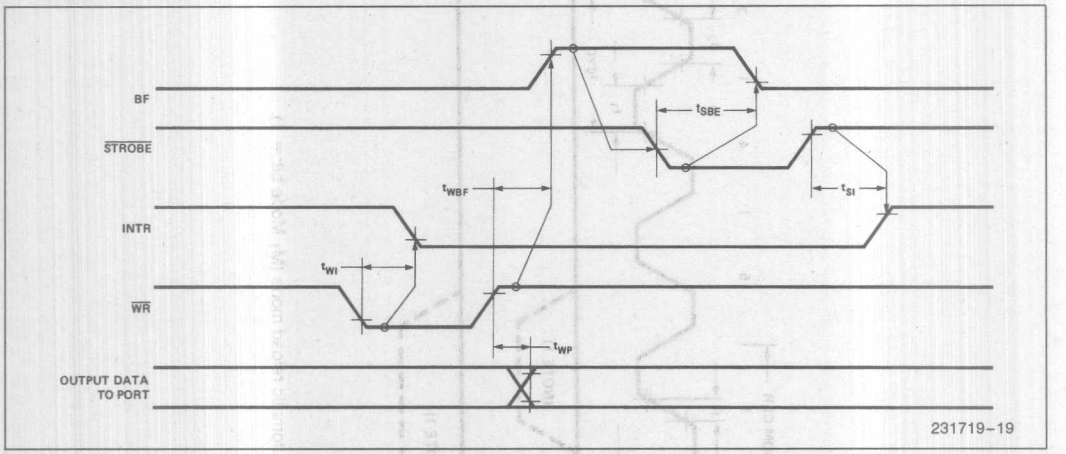


WAVEFORMS (Continued)

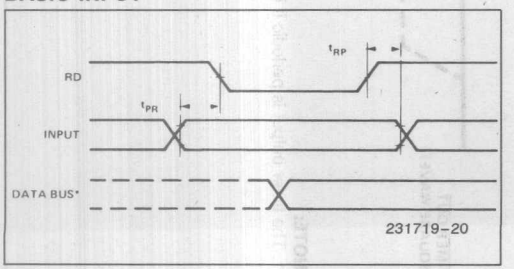
STROBED INPUT



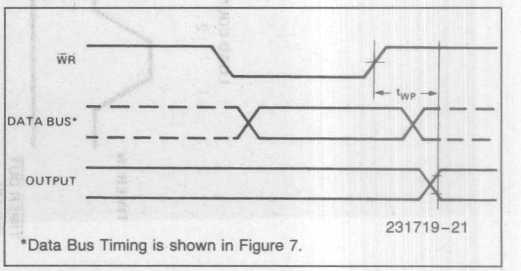
STROBED OUTPUT



BASIC INPUT



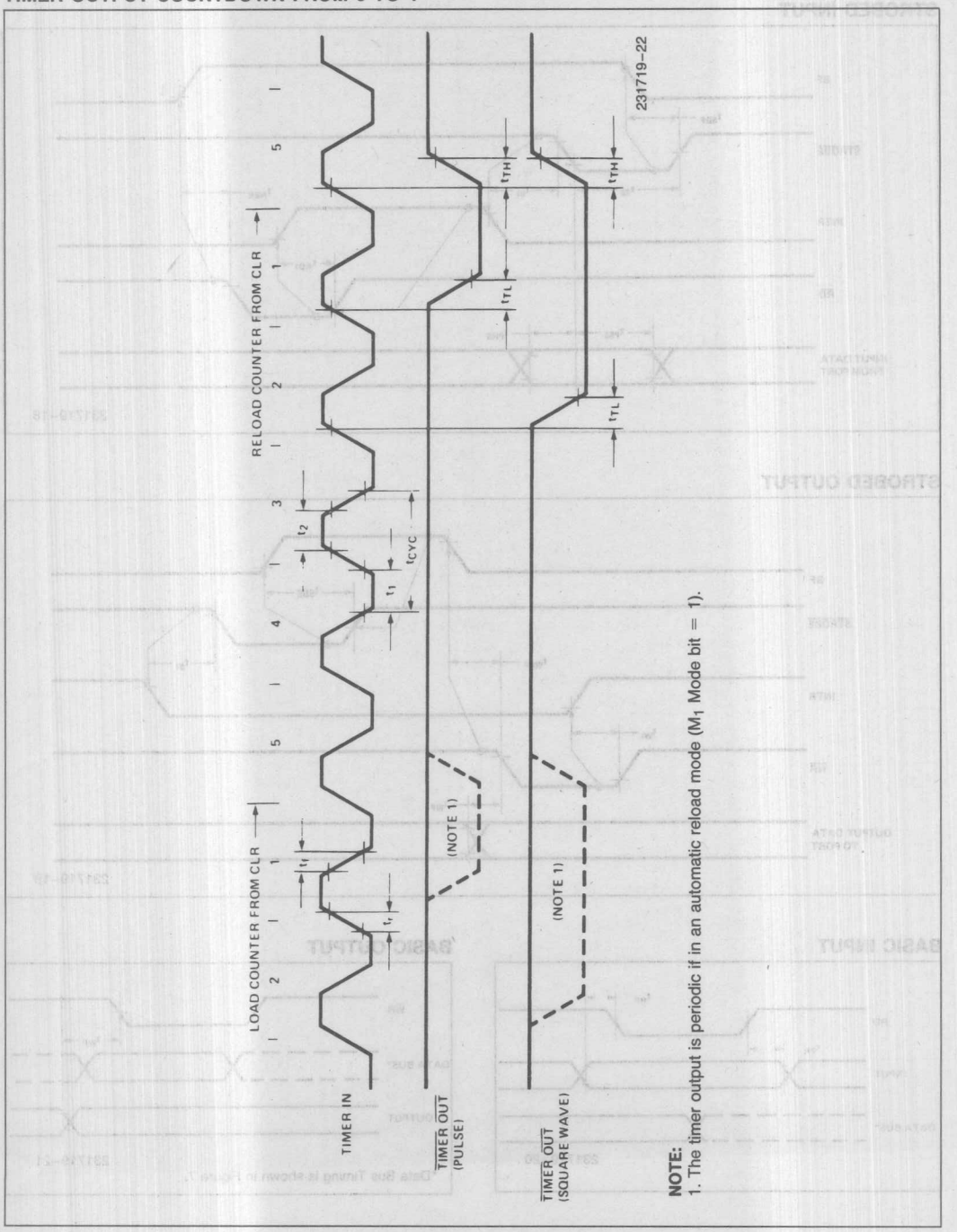
BASIC OUTPUT



*Data Bus Timing is shown in Figure 7.

WAVEFORMS (Continued)

TIMER OUTPUT COUNTDOWN FROM 5 TO 1



NOTE:
1. The timer output is periodic if in an automatic reload mode (M_1 Mode bit = 1).

8185/8185-2
1024 x 8-BIT STATIC RAM FOR MCS®-85

- **Multiplexed Address and Data Bus**
- **Directly Compatible with 8085AH and 8088 Microprocessors**
- **Low Operating Power Dissipation**
- **Low Standby Power Dissipation**
- **Single +5V Supply**
- **High Density 18-Pin Package**

The Intel 8185 is an 8192-bit static random access memory (RAM) organized as 1024 words by 8-bits using N-channel Silicon-Gate MOS technology. The multiplexed address and data bus allows the 8185 to interface directly to the 8085AH and 8088 microprocessors to provide a maximum level of system integration.

The low standby power dissipation minimizes system power requirements when the 8185 is disabled.

The 8185-2 is a high-speed selected version of the 8185 that is compatible with the 5 MHz 8085AH-2 and the 5 MHz 8088.

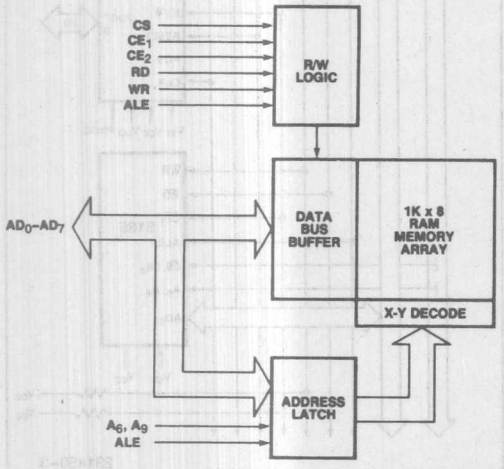


Figure 1. Block Diagram

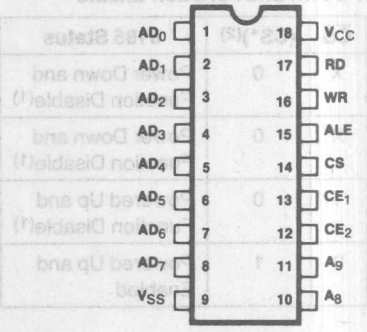


Figure 2. Pin Configuration

Pin Names

AD_0–AD_7	Address/Data Lines
A_8, A_9	Address Lines
CS	Chip Select
CE_1	Chip Enable (IO/M)
CE_2	Chip Enable
ALE	Address Latch Enable
WR	Write Enable

May 1987
Order Number: 231450-002

FUNCTIONAL DESCRIPTION

The 8185 has been designed to provide for direct interface to the multiplexed bus structure and bus timing of the 8085A microprocessor.

At the beginning of an 8185 memory access cycle, the 8-bit address on AD_{0-7}, A_8 and A_9, and the status of $\overline{CE}_1$ and CE_2 are all latched internally in the 8185 by the falling edge of ALE. If the latched status of both $\overline{CE}_1$ and CE_2 are active, the 8185 powers itself up, but no action occurs until the $\overline{CS}$ line goes low and the appropriate $\overline{RD}$ or $\overline{WR}$ control signal input is activated.

The $\overline{CS}$ input is not latched by the 8185 in order to allow the maximum amount of time for address decoding in selecting the 8185 chip. Maximum power consumption savings will occur, however, only when $\overline{CE}_1$ and CE_2 are activated selectively to power down the 8185 when it is not in use. A possible connection would be to wire the 8085A's IO/$\overline{M}$ line to the 8185's $\overline{CE}_1$ input, thereby keeping the 8185 powered down during I/O and interrupt cycles.

Table 1. Truth Table for Power Down and Function Enable

$\overline{CE}_1$	CE_2	$\overline{CS}$	(CS*)(2)	8185 Status
1	X	X	0	Power Down and Function Disable(1)
X	0	X	0	Power Down and Function Disable(1)
0	1	1	0	Powered Up and Function Disable(1)
0	1	0	1	Powered Up and Enabled

NOTES:
X = Don't Care.
1: Function Disable implies Data Bus in high impedance state and not writing.
2: CS* = $(\overline{CE}_1 = 0) \times (CE_2 = 1) \times (\overline{CS} = 0)$.
CS* = 1 signifies all chip enables and chip select active.

Table 2. Truth Table for Control and Data Bus Pin Status

(CS*)	$\overline{RD}$	$\overline{WR}$	AD_{0-7} During Data Portion of Cycle	8185 Function
0	X	X	Hi-Impedance	No Function
1	0	1	Data from Memory	Read
1	1	0	Data to Memory	Write
1	1	1	Hi-Impedance	Reading, but not Driving Data Bus

NOTE:
X = Don't Care.

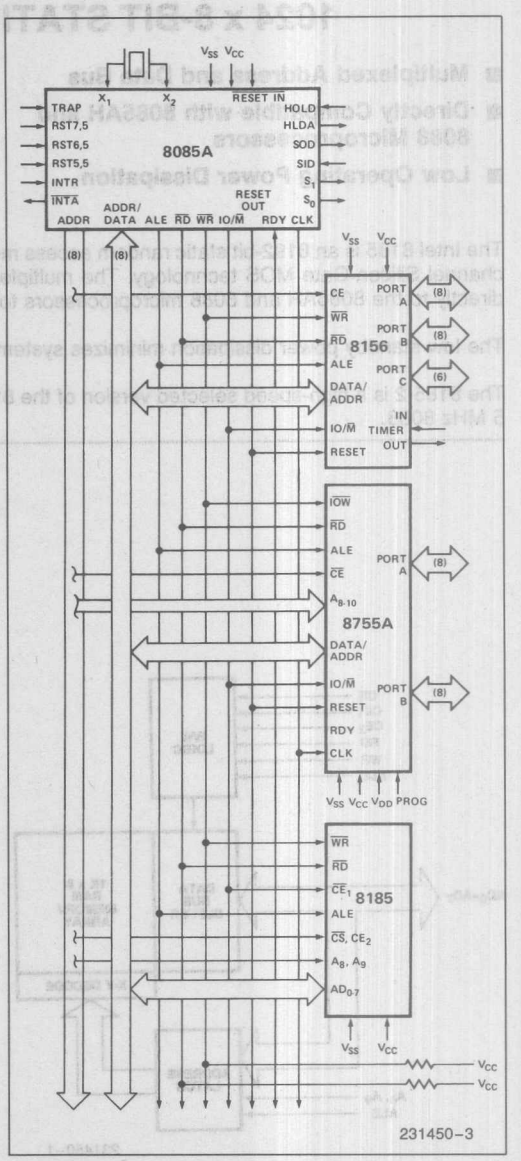


Figure 3. 8185 in an MCS®-85 System

4 Chips:
2K Bytes EPROM
1.25K Bytes RAM
38 I/O Lines
1 Counter/Timer
2 Serial I/O Lines
5 Interrupt Inputs

iAPX 88 FIVE CHIP SYSTEM:

- 1.25K Bytes RAM
- 2K Bytes EPROM
- 38 I/O Pins
- 1 Internal Timer
- 2 Interrupt Levels

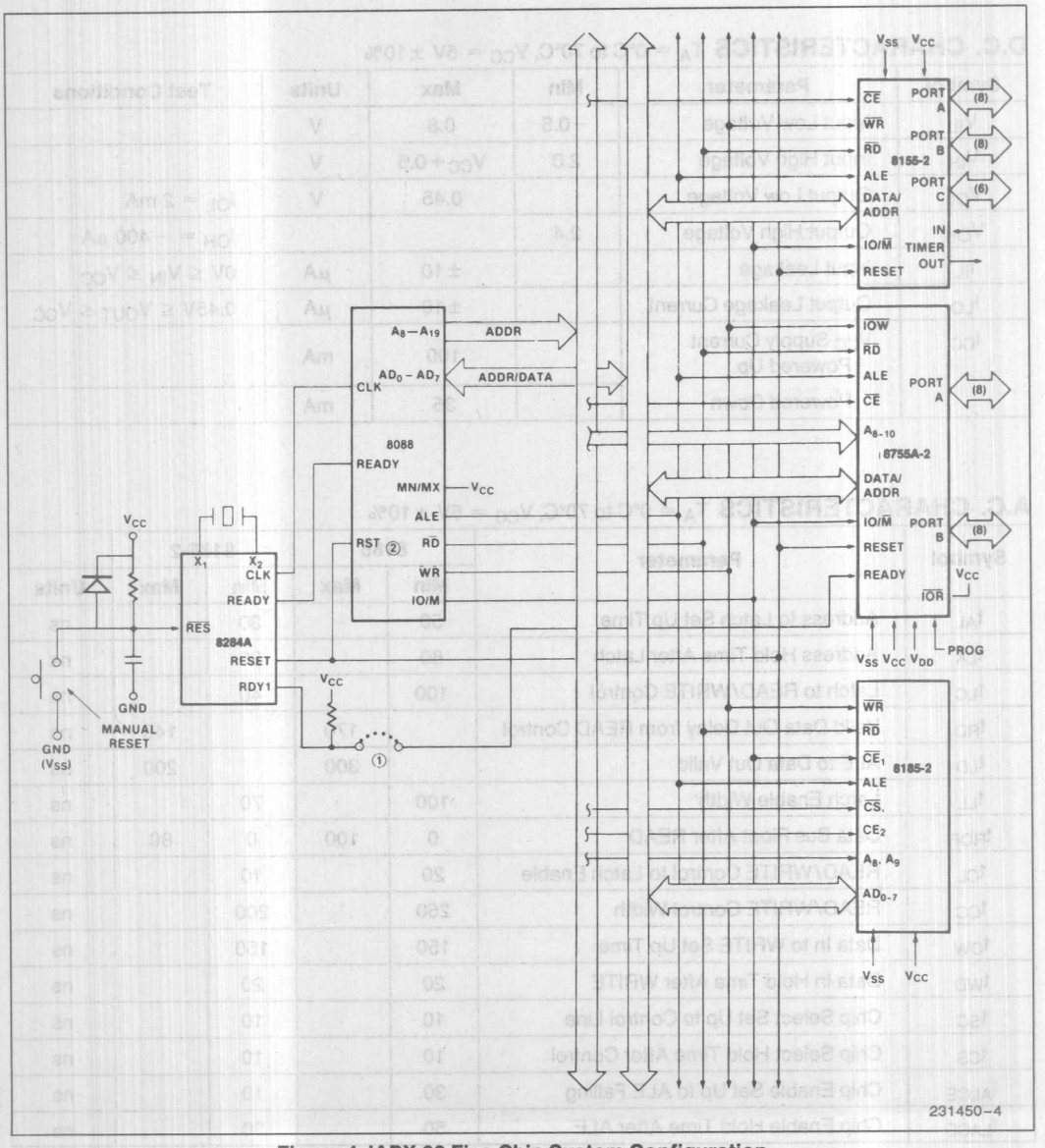


Figure 4. iAPX 88 Five Chip System Configuration

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias .....0°C to +70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin
with Respect to Ground.....−0.5V to +7V
Power Dissipation.....1.5W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

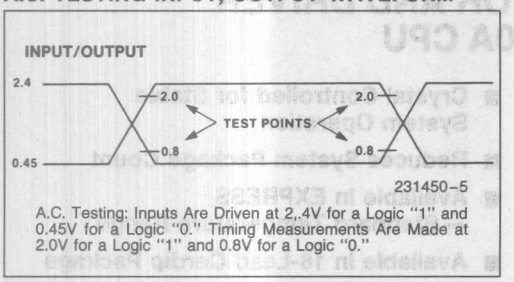
D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4			I_{OH} = −400 μA
I_{IL}	Input Leakage		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current		±10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	V_{CC} Supply Current Powered Up		100	mA	
	Powered Down		35	mA	

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±10%

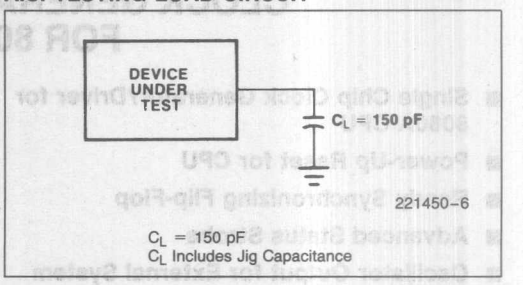
Symbol	Parameter	8185		8185-2		
		Min	Max	Min	Max	Units
t_{AL}	Address to Latch Set Up Time	50		30		ns
t_{LA}	Address Hold Time After Latch	80		30		ns
t_{LC}	Latch to READ/WRITE Control	100		40		ns
t_{RD}	Valid Data Out Delay from READ Control		170		140	ns
t_{LD}	ALE to Data Out Valid		300		200	ns
t_{LL}	Latch Enable Width	100		70		ns
t_{RDF}	Data Bus Float After READ	0	100	0	80	ns
t_{CL}	READ/WRITE Control to Latch Enable	20		10		ns
t_{CC}	READ/WRITE Control Width	250		200		ns
t_{DW}	Data In to WRITE Set Up Time	150		150		ns
t_{WD}	Data In Hold Time After WRITE	20		20		ns
t_{SC}	Chip Select Set Up to Control Line	10		10		ns
t_{CS}	Chip Select Hold Time After Control	10		10		ns
t_{ALCE}	Chip Enable Set Up to ALE Falling	30		10		ns
t_{LACE}	Chip Enable Hold Time After ALE	50		30		ns

A.C. TESTING INPUT, OUTPUT WAVEFORM



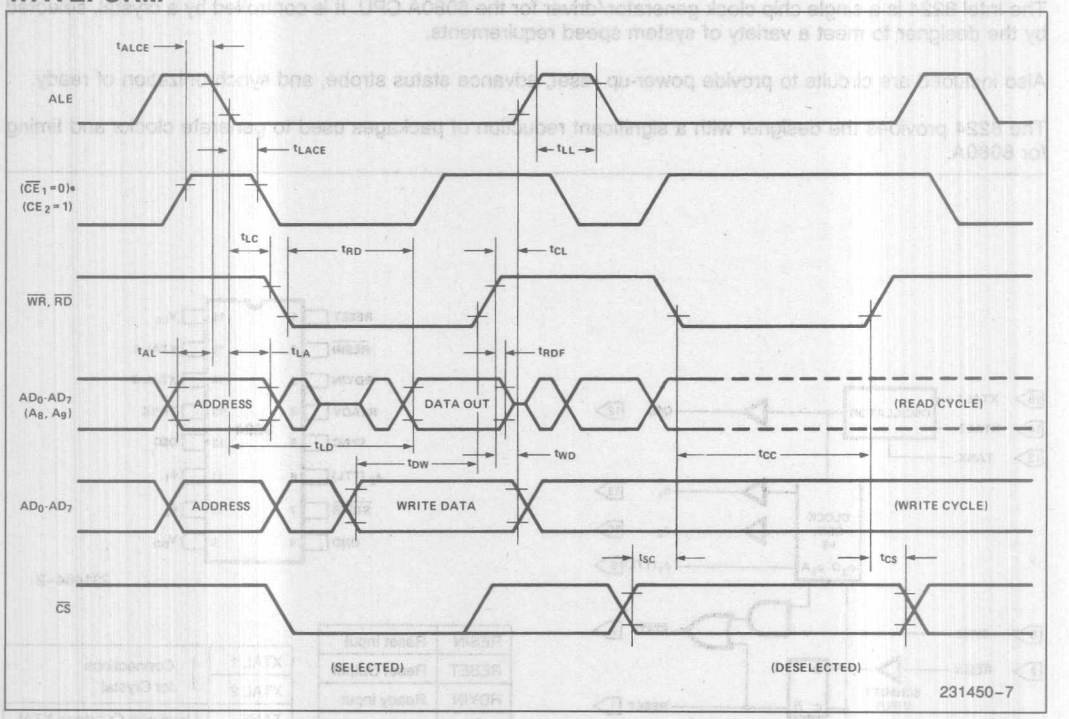
A.C. Testing: Inputs Are Driven at 2,.4V for a Logic "1" and 0.45V for a Logic "0." Timing Measurements Are Made at 2.0V for a Logic "1" and 0.8V for a Logic "0."

A.C. TESTING LOAD CIRCUIT



C_L = 150 pF
C_L Includes Jig Capacitance

WAVEFORM



8224
CLOCK GENERATOR AND DRIVER FOR 8080A CPU

- **Single Chip Clock Generator/Driver for 8080A CPU**
- **Power-Up Reset for CPU**
- **Ready Synchronizing Flip-Flop**
- **Advanced Status Strobe**
- **Oscillator Output for External System Timing**
- **Crystal Controlled for Stable System Operation**
- **Reduces System Package Count**
- **Available in EXPRESS — Standard Temperature Range**
- **Available in 16-Lead Cerdip Package**

(See Packaging Spec, Order #231369)

The Intel 8224 is a single chip clock generator/driver for the 8080A CPU. It is controlled by a crystal, selected by the designer to meet a variety of system speed requirements.

Also included are circuits to provide power-up reset, advance status strobe, and synchronization of ready.

The 8224 provides the designer with a significant reduction of packages used to generate clocks and timing for 8080A.

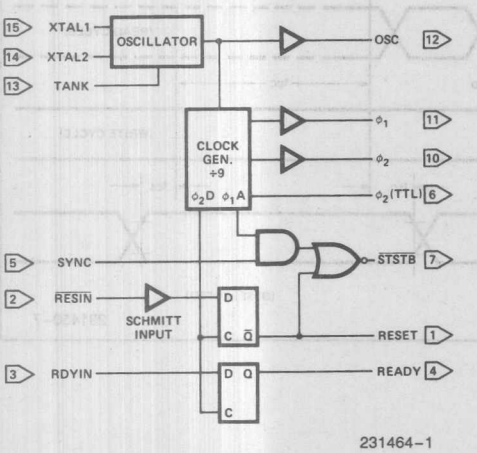
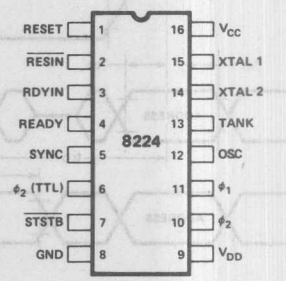


Figure 1. Block Diagram



RESIN	Reset Input
RESET	Reset Output
RDYIN	Ready Input
READY	Ready Output
SYNC	Sync Input
STSTB	Status STB (Active Low)
ϕ_1, ϕ_2	8080 Clocks

XTAL 1, XTAL 2	Connections for Crystal
TANK	Used with Overtone XTAL
OSC	Oscillator Output
ϕ_2 (TTL)	ϕ_2 CLK (TTL Level)
V_{CC}	+5V
V_{DD}	+12V
GND	0V

Figure 2. Pin Configuration

December 1986
Order Number: 231464-001

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias .....0°C to +70°C
Storage Temperature .....−65°C to +150°C
Supply Voltage, V_{CC} .....−0.5V to +7V
Supply Voltage, V_{DD} .....−0.5V to +13.5V
Input Voltage .....−1.5V to +7V
Output Current .....100 mA

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to +70°C, V_{CC} = +5.0V ±5%, V_{DD} = +12V ±5%

Symbol	Parameter	Limits			Units	Test Conditions
		Min	Typ	Max		
I_F	Input Current Loading			−0.25	mA	V_F = 0.45V
I_R	Input Leakage Current			10	μA	V_R = 5.25V
V_C	Input Forward Clamp Voltage			1.0	V	I_C = −5 mA
V_{IL}	Input "Low" Voltage			0.8	V	V_{CC} = 5.0V
V_{IH}	Input "High" Voltage	2.6			V	Reset Input
		2.0			V	All Other Inputs
$V_{IH}-V_{IL}$	RESIN Input Hysteresis	0.25			V	V_{CC} = 5.0V
V_{OL}	Output "Low" Voltage			0.45	V	(ϕ_1, ϕ_2), Ready, Reset, $\overline{STSTB}$ I_{OL} = 2.5 mA
				0.45	V	All Other Outputs I_{OL} = 15 mA
V_{OH}	Output "High" Voltage ϕ_1, ϕ_2	9.4			V	I_{OH} = −100 μA
	READY, RESET	3.6			V	I_{OH} = −100 μA
	All Other Outputs	2.4			V	I_{OH} = −1 mA
I_{CC}	Power Supply Current			115	mA	
I_{DD}	Power Supply Current			12	mA	

NOTE:
1. For crystal frequencies of 18 MHz connect 510Ω resistors between the X1 input and ground as well as the X2 input and ground to prevent oscillation at harmonic frequencies.

Crystal Requirements

Tolerance: 0.005% at 0°C–70°C
Resonance: Series (Fundamental)*
Load Capacitance: 20 pF–35 pF
Equivalent Resistance: 75Ω–20Ω

Power Dissipation (Min): 4 mW

***NOTE:**
With tank circuit use 3rd overtone mode.

A.C. CHARACTERISTICS

Symbol	Parameter	Limits			Units	Test Conditions
		Min	Typ	Max		
$t_{\phi 1}$	ϕ_1 Pulse Width	$\frac{2tcy}{9} - 20$ ns			ns	C_L = 20 pF to 50 pF
$t_{\phi 2}$	ϕ_2 Pulse Width	$\frac{5tcy}{9} - 35$ ns				
t_{D1}	ϕ_1 to ϕ_2 Delay	0				
t_{D2}	ϕ_2 to ϕ_1 Delay	$\frac{2tcy}{9} - 14$ ns				
t_{D3}	ϕ_1 to ϕ_2 Delay	$\frac{2tcy}{9}$		$\frac{2tcy}{9} + 20$ ns		
t_R	ϕ_1 and ϕ_2 Rise Time			20		
t_F	ϕ_1 and ϕ_2 Fall Time			20		
$t_{D\phi 2}$	ϕ_2 to ϕ_2 (TTL) Delay	−5		+15	ns	ϕ_2 TTL, CL = 30 R_1 = 300Ω R_2 = 600Ω
t_{DSS}	ϕ_2 to $\overline{STSTB}$ Delay	$\frac{6tcy}{9} - 30$ ns		$\frac{6tcy}{9}$	ns	
t_{PW}	$\overline{STSTB}$ Pulse Width	$\frac{tcy}{9} - 15$ ns			ns	$\overline{STSTB}$, C_L = 15 pF R_1 = 2K R_2 = 4K
t_{DRS}	RDYIN Setup Time to Status Strobe	50 ns − $\frac{4tcy}{9}$				
t_{DRH}	RDYIN Hold Time after $\overline{STSTB}$	$\frac{4tcy}{9}$				
t_{DR}	RDYIN or RESIN to ϕ_2 Delay	$\frac{4tcy}{9} - 25$ ns			ns	Ready & Reset C_L = 10 pF R_1 = 2K R_2 = 4K
t_{CLK}	CLK Period		$\frac{tcy}{9}$		ns	
f_{max}	Maximum Oscillating Frequency			27	MHz	
C_{in}	Input Capacitance			8	pF	V_{CC} = +5.0V V_{DD} = +12V V_{BIAS} = 2.5V f = 1 MHz

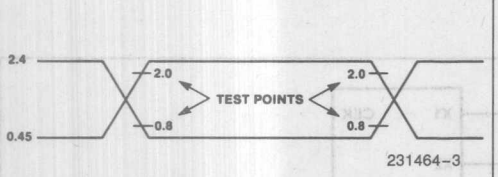
NOTE:
These formulas are based on the internal workings of the part and intended for customer convenience. Actual testing of the part is done at t_{cy} = 488.28 ns.

A.C. CHARACTERISTICS (Continued)

For t_{CY} = 488.28 ns; T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{DD} = +12V ±5%

Symbol	Parameter	Limits			Units	Test Conditions
		Min	Typ	Max		
$t_{\phi 1}$	ϕ_1 Pulse Width	89			ns	t_{CY} = 488.28 ns
$t_{\phi 2}$	ϕ_2 Pulse Width	236			ns	
t_{D1}	Delay ϕ_1 to ϕ_2	0			ns	
t_{D2}	Delay ϕ_2 to ϕ_1	95			ns	ϕ_1 & ϕ_2 Loaded to C_L = 20 pF to 50 pF
t_{D3}	Delay ϕ_1 to ϕ_2 Leading Edges	109		129	ns	
t_r	Output Rise Time			20	ns	
t_f	Output Fall Time			20	ns	
t_{DSS}	ϕ_2 to $\overline{STSTB}$ Delay	296		326	ns	
$t_{D\phi 2}$	ϕ_2 to ϕ_2 (TTL) Delay	−5		+15	ns	
t_{PW}	Status Strobe Pulse Width	40			ns	Ready & Reset Loaded to 2 mA/10 pF All measurements referenced to 1.5V unless specified otherwise.
t_{DRS}	RDYIN Setup Time to $\overline{STSTB}$	−167			ns	
t_{DRH}	RDYIN Hold Time after $\overline{STSTB}$	217			ns	
t_{DR}	READY or RESET to ϕ_2 Delay	192			ns	
f_{MAX}	Oscillator Frequency			18.432	MHz	

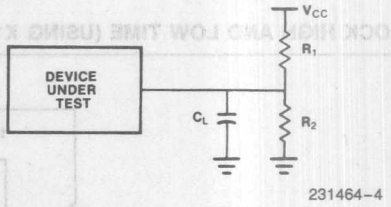
A.C. TESTING, INPUT, OUTPUT WAVEFORM



231464-3

A.C. Testing: Inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0" (unless otherwise noted).

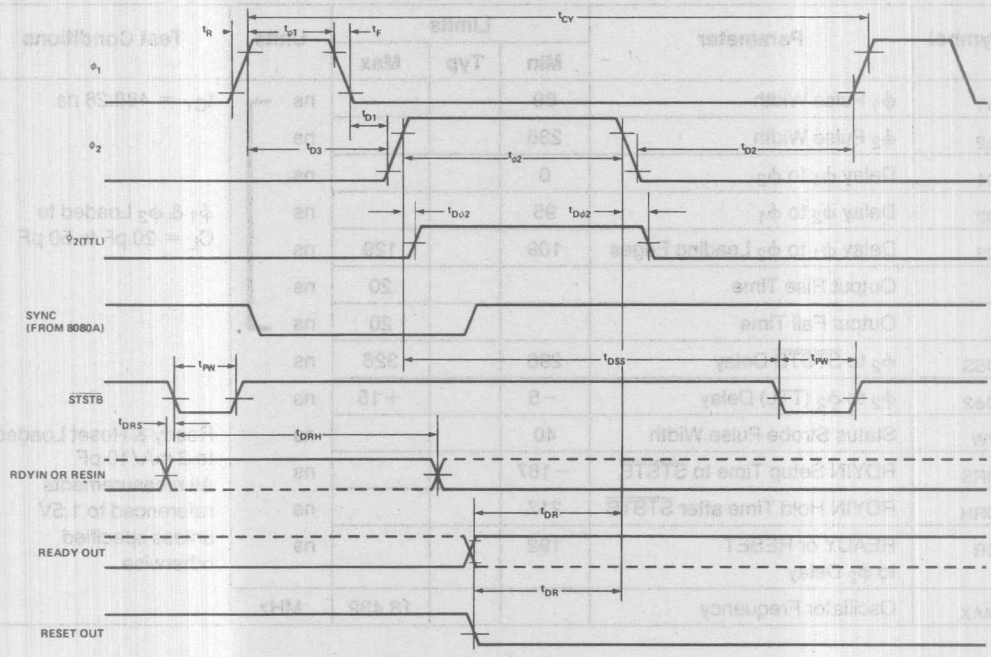
A.C. TESTING LOAD CIRCUIT



231464-4

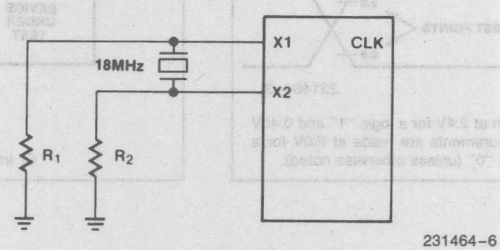
C_L Includes Jig Capacitance

WAVEFORMS



VOLTAGE MEASUREMENT POINTS: ϕ_1, ϕ_2 Logic "0" = 1.0V, Logic "1" = 8.0V. All other signals measured at 1.5V.

CLOCK HIGH AND LOW TIME (USING X1, X2)



8228
SYSTEM CONTROLLER AND BUS DRIVER FOR 8080A CPU

- **Single Chip System Control for MCS®-80 Systems**
- **Built-In Bidirectional Bus Driver for Data Bus Isolation**
- **Allows the Use of Multiple Byte Instructions (e.g. CALL) for Interrupt Acknowledge**
- **Reduces System Package Count**
- **User Selected Single Level Interrupt Vector (RST 7)**
- **Available in EXPRESS — Standard Temperature Range**
- **Available in 28-Lead Cerdip and Plastic Packages**

(See Packaging Spec, Order #231369)

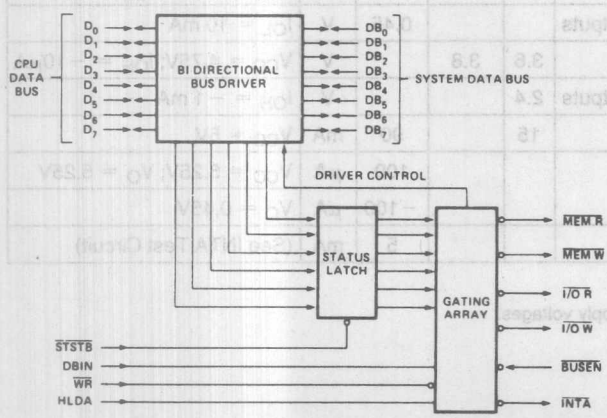
The Intel® 8228 is a single chip system controller and bus driver for MCS®-80. It generates all signals required to directly interface MCS-80 family RAM, ROM, and I/O components.

A bidirectional bus driver is included to provide high system TTL fan-out. It also provides isolation of the 8080 data bus from memory and I/O. This allows for the optimization of control signals, enabling the systems designer to use slower memory and I/O. The isolation of the bus driver also provides for enhanced system noise immunity.

A user selected single level interrupt vector (RST 7) is provided to simplify real time, interrupt driven, small system requirements. The 8228 also generates the correct control signals to allow the use of multiple byte instructions (e.g., CALL) in response to an interrupt acknowledge by the 8080A. This feature permits large, interrupt driven systems to have an unlimited number of interrupt levels.

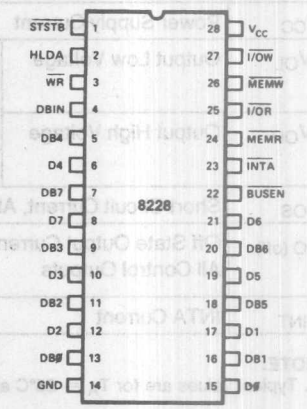
The 8228 is designed to support a wide variety of system bus structures and also reduce system package count for cost effective, reliable design of MCS-80 systems.

NOTE:
The specifications for the 3228 are identical with those for the 8228.



231465-1

Figure 1. Block Diagram



231465-2

D7-D0	Data Bus (8080 Side)
DB7-DB0	Data Bus (System Side)
I/$\overline{OR}$	I/O Read
I/$\overline{OW}$	I/O Write
$\overline{MEMR}$	Memory Read
$\overline{MEMW}$	Memory Write
DBIN	DBIN (from 8080)

INTA	Interrupt Acknowledge
HLDA	HLDA (from 8080)
$\overline{WR}$	WR (from 8080)
$\overline{BUSEN}$	Bus Enable Input
$\overline{STSTB}$	Status Strobe (from 8224)
V_{CC}	+5V
GND	0 Volts

Figure 2. Pin Configuration

September 1987
Order Number: 231465-002

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias .....0°C to +70°C
Storage Temperature .....−65°C to +150°C
Supply Voltage, V_{CC} .....−0.5V to +7V
Input Voltage .....−1.5 to +7V
Output Current .....100 mA

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$

Symbol	Parameter		Limits			Unit	Test Conditions
			Min	Typ(1)	Max		
V_C	Input Clamp Voltage, All Input			0.75	−1.0	V	$V_{CC} = 4.75V$; $I_C = -5$ mA
I_F	Input Load Current	$\overline{STSTB}$			500	μA	$V_{CC} = 5.25V$
		D_2 & D_6			750	μA	$V_F = 0.45V$
		D_0, D_1, D_4, D_5 & D_7			250	μA	
		All Other Inputs			250	μA	
I_R	Input Leakage Current	$\overline{STSTB}$			100	μA	$V_{CC} = 5.25V$ $V_R = 5.25V$
		DB_0–DB_7			20	μA	
		All Other Inputs			100	μA	
V_{TH}	Input Threshold Voltage, All Inputs		0.8		2.0	V	$V_{CC} = 5V$
I_{CC}	Power Supply Current			140	190	mA	$V_{CC} = 5.25V$
V_{OL}	Output Low Voltage	D_0–D_7			0.45	V	$V_{CC} = 4.75V$; $I_{OL} = 2$ mA
		All Other Outputs			0.45	V	$I_{OL} = 10$ mA
V_{OH}	Output High Voltage	D_0–D_7	3.6	3.8		V	$V_{CC} = 4.75V$; $I_{OH} = -10\mu A$
		All Other Outputs	2.4			V	$I_{OH} = -1$ mA
I_{OS}	Short Circuit Current, All Outputs		15		90	mA	$V_{CC} = 5V$
$I_{O(off)}$	Off State Output Current All Control Outputs				100	μA	$V_{CC} = 5.25V$; $V_O = 5.25V$
					−100	μA	$V_O = 0.45V$
I_{INT}	INTA Current				5	mA	(See INTA Test Circuit)

NOTE:
1. Typical values are for $T_A = 25°C$ and nominal supply voltages.

CAPACITANCE V_{BIAS} = 2.5V, V_{CC} = 5.0V, T_A = 25°C, f = 1 MHz

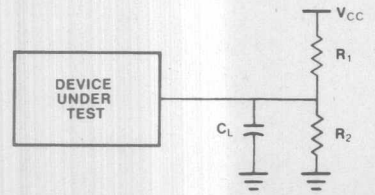
1. This parameter is periodically sampled and not 100% tested.

Symbol	Parameter	Limits			Unit
		Min	Typ(1)	Max	
C_{IN}	Input Capacitance		8	12	pF
C_{OUT}	Output Capacitance Control Signals		7	15	pF
I/O	I/O Capacitance (D or DB)		8	15	pF

A.C. CHARACTERISTICS T_A = 0°C to +70°C, V_{CC} = 5V ±5%

Symbol	Parameter	Limits		Unit	Conditions
		Min	Max		
t_{PW}	Width of Status Strobe	22		ns	
t_{SS}	Setup Time, Status Inputs D_0–D_7	8		ns	
t_{SH}	Hold Time, Status Inputs D_0–D_7	5		ns	
t_{DC}	Delay from $\overline{STSTB}$ to any Control Signal	20	60	ns	C_L = 100 pF
t_{RR}	Delay from DBIN to Control Outputs		30	ns	C_L = 100 pF
t_{RE}	Delay from DBIN to Enable/Disable 8080 Bus		45	ns	C_L = 25 pF
t_{RD}	Delay from System Bus to 8080 Bus during Read		30	ns	C_L = 25 pF
t_{WR}	Delay from $\overline{WR}$ to Control Outputs	5	45	ns	C_L = 100 pF
t_{WE}	Delay to Enable System Bus DB_0–DB_7 after $\overline{STSTB}$		30	ns	C_L = 100 pF
t_{WD}	Delay from 8080 Bus D_0–D_7 to System Bus DB_0–DB_7 during Write	5	40	ns	C_L = 100 pF
t_E	Delay from $\overline{\text{System Bus Enable}}$ to System Bus DB_0–DB_7		30	ns	C_L = 100 pF
t_{HD}	HLDA to Read Status Outputs		25	ns	
t_{DS}	Setup Time, System Bus Inputs to HLDA	10		ns	
t_{DH}	Hold Time, System Bus Inputs to HLDA	20		ns	C_L = 100 pF

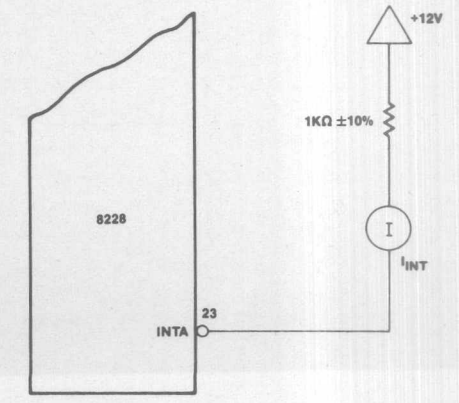
AC TESTING LOAD CIRCUIT



231465-3

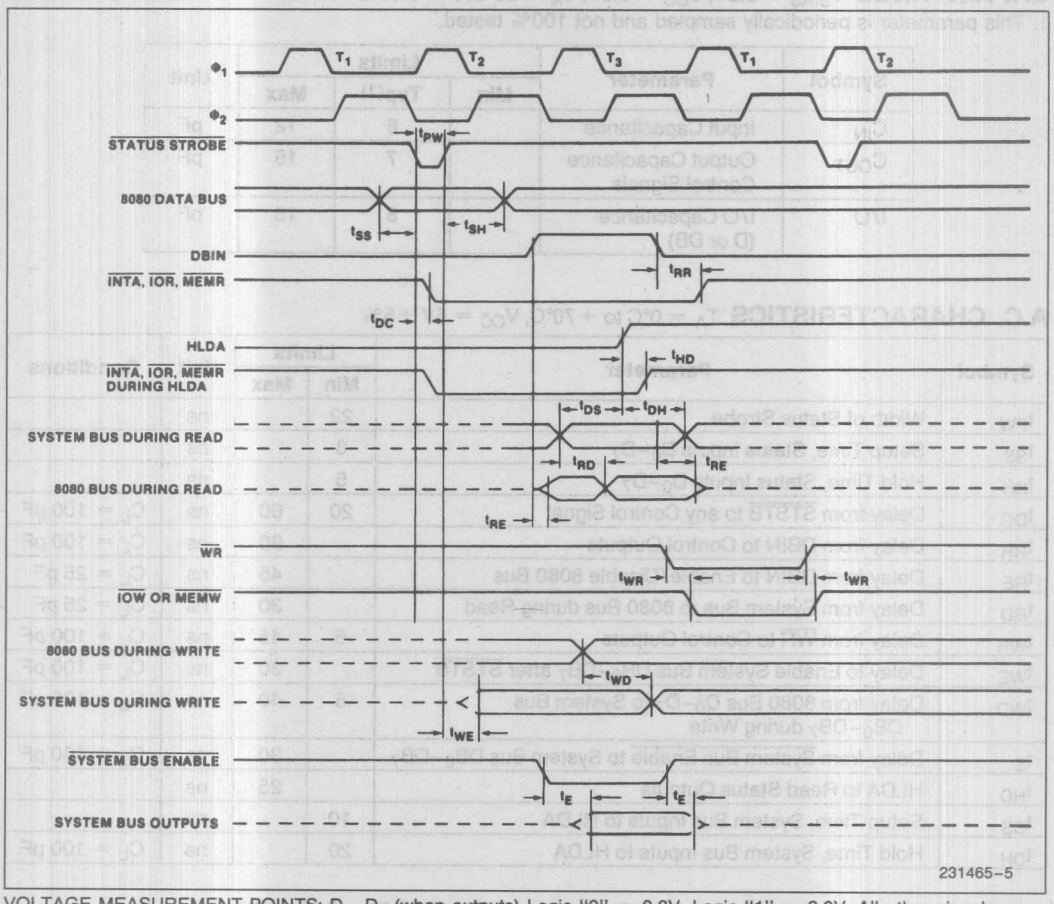
For D_0–D_7; R_1 = 4 KΩ, R_2 = ∞Ω, C_L = 25 pF.
For all other outputs: R_1 = 500Ω, R_2 = 1 KΩ, C_L = 100 pF.

INTA Test Circuit (for RST 7)



231465-4

WAVEFORMS



VOLTAGE MEASUREMENT POINTS: D_0–D_7 (when outputs) Logic "0" = 0.8V, Logic "1" = 3.0V. All other signals measured at 1.5V.

8755A
16,384-BIT EPROM WITH I/O

- 2048 Words x 8 Bits
- Single +5V Power Supply (V_{CC})
- Directly Compatible with 8085AH
- U.V. Erasable and Electrically Reprogrammable
- Internal Address Latch
- 2 General Purpose 8-Bit I/O Ports
- Each I/O Port Line Individually Programmable as Input or Output
- Multiplexed Address and Data Bus
- 40-Pin DIP
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel 8755A is an erasable and electrically reprogrammable ROM (EPROM) and I/O chip to be used in the 8085AH microprocessor systems. The EPROM portion is organized as 2048 words by 8 bits. It has a maximum access time of 450 ns to permit use with no wait states in an 8085AH CPU.

The I/O portion consists of 2 general purpose I/O ports. Each I/O port has 8 port lines, and each I/O port line is individually programmable as input or output.

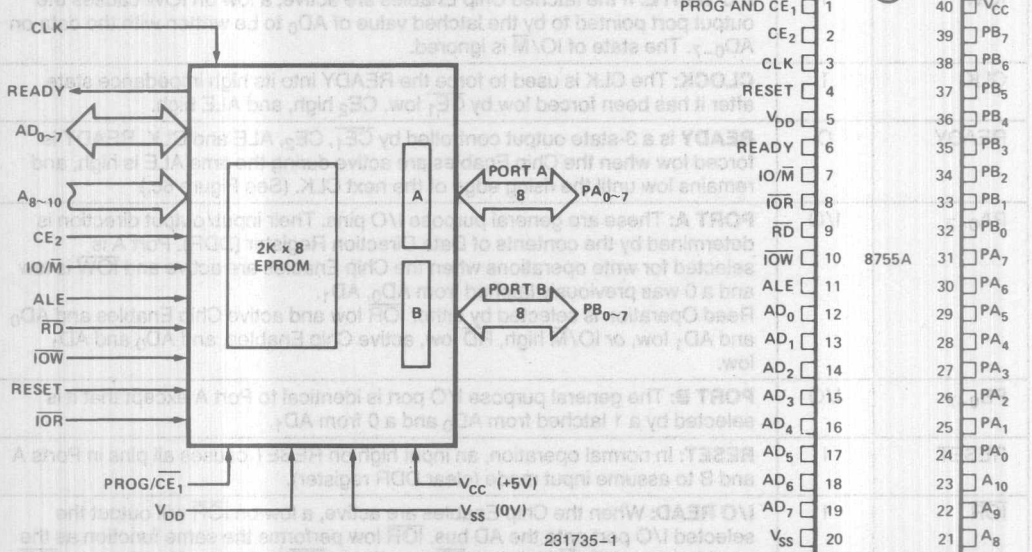


Figure 1. Block Diagram

231735-2

Figure 2. Pin Configuration

November 1986
Order Number: 231735-002

Table 1. Pin Description

Symbol	Type	Name and Function
ALE	I	**ADDRESS LATCH ENABLE:** When Address Latch Enable goes *high,* AD_{0-7}, IO/$\overline{M}$, A_{8-10}, CE_2, and $\overline{CE_1}$ enter the address latches. The signals, (AD, IO/$\overline{M}$, AD_{8-10}, CE_2, $\overline{CE_1}$) are latched in at the trailing edge of ALE.
AD_{0-7}	I	**BIDIRECTIONAL ADDRESS/DATA BUS:** The lower 8 bits of the PROM or I/O address are applied to the bus lines when ALE is high. During an I/O cycle, Port A or B is selected based on the latched value of AD_O. IF $\overline{RD}$ or $\overline{IOR}$ is low when the latched Chip Enables are active, the output buffers present data on the bus.
AD_{8-10}	I	**ADDRESS BUS:** These are the high order bits of the PROM address. They do not affect I/O operations.
PROG/$\overline{CE_1}$ CE_2	I	**CHIP ENABLE INPUTS:** $\overline{CE_1}$ is active low and CE_2 is active high. The 8755A can be accessed only when *both* Chip Enables are active at the time the ALE signal latches them up. If either Chip Enable input is not active, the AD_{0-7}, and READY ouputs will be in a high impedance state. $\overline{CE_1}$ is also used as a programming pin. (See section on programming.)
IO/$\overline{M}$	I	**I/O MEMORY:** If the latched IO/$\overline{M}$ is high when $\overline{RD}$ is low, the output data comes from an I/O port. If it is low the output data comes from the PROM.
$\overline{RD}$	I	**READ:** If the latched Chip Enables are active when $\overline{RD}$ goes low, the AD_{0-7} output buffers are enabled and output either the selected PROM location or I/O port. When both $\overline{RD}$ and $\overline{IOR}$ are high, the AD_{0-7} output buffers are 3-stated.
$\overline{IOW}$	I	**I/O WRITE:** If the latched Chip Enables are active, a low on $\overline{IOW}$ causes the output port pointed to by the latched value of AD_0 to be written with the data on AD_{0-7}. The state of IO/$\overline{M}$ is ignored.
CLK	I	**CLOCK:** The CLK is used to force the READY into its high impedance state after it has been forced low by $\overline{CE_1}$ low, CE_2 high, and ALE high.
READY	O	**READY** is a 3-state output controlled by $\overline{CE_1}$, CE_2, ALE and CLK. READY is forced low when the Chip Enables are active during the time ALE is high, and remains low until the rising edge of the next CLK. (See Figure 6c.)
PA_{0-7}	I/O	**PORT A:** These are general purpose I/O pins. Their input/output direction is determined by the contents of Data Direction Register (DDR). Port A is selected for write operations when the Chip Enables are active and $\overline{IOW}$ is low and a 0 was previously latched from AD_0, AD_1. Read Operation is selected by either $\overline{IOR}$ low and active Chip Enables and AD_0 and AD_1 low, *or* IO/$\overline{M}$ high, $\overline{RD}$ low, active Chip Enables, and AD_0 and AD_1 low.
PB_{0-7}	I/O	**PORT B:** The general purpose I/O port is identical to Port A except that it is selected by a 1 latched from AD_0 and a 0 from AD_1.
RESET	I	**RESET:** In normal operation, an input high on RESET causes all pins in Ports A and B to assume input mode (clear DDR register).
$\overline{IOR}$	I	**I/O READ:** When the Chip Enables are active, a low on $\overline{IOR}$ will output the selected I/O port onto the AD bus. $\overline{IOR}$ low performs the same function as the combination of IO/$\overline{M}$ high and $\overline{RD}$ low. When $\overline{IOR}$ is not used in a system, $\overline{IOR}$ should be tied to V_{CC} ("1").
V_{CC}		**POWER:** +5V supply.
V_{SS}		**GROUND:** Reference.
V_{DD}		**POWER SUPPLY:** V_{DD} is a programming voltage, and must be tied to V_{CC} when the 8755A is being read. For programming, a high voltage is supplied with V_{DD} = 25V, typical. (See section on programming.)

FUNCTIONAL DESCRIPTION

PROM Section

The 8755A contains an 8-bit address latch which allows it to interface directly to MCS®-48 and MCS®-85 processors without additional hardware.

The PROM section of the chip is addressed by the 11-bit address and the Chip Enables. The address, $\overline{CE_1}$ and CE_2 are latched into the address latches on the falling edge of ALE. If the latched Chip Enables are active and IO/$\overline{M}$ is low when $\overline{RD}$ goes low, the contents of the PROM location addressed by the latched address are put out on the AD_{0-7} lines (provided that V_{DD} is tied to V_{CC}).

I/O Section

The I/O section of the chip is addressed by the latched value of AD_{0-1}. Two 8-bit Data Direction Registers (DDR) in 8755A determine the input/output status of each pin in the corresponding ports. A "0" in a particular bit position of a DDR signifies that the corresponding I/O port bit is in the input mode. A "1" in a particular bit position signifies that the corresponding I/O port bit is in the output mode. In this manner the I/O ports of the 8755A are bit-by-bit programmable as inputs or outputs. The table summarizes port and DDR designation. DDR's cannot be read.

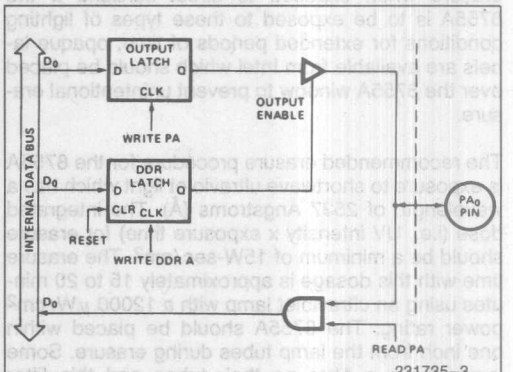
AD_1	AD_0	Selection
0	0	Port A
0	1	Port B
1	0	Port A Data Direction Register (DDR A)
1	1	Port B Data Direction Register (DDR B)

When $\overline{IOW}$ goes low and the Chip Enables are active, the data on the AD_{0-7} is written into I/O port selected by the latched value of AD_{0-1}. During this operation all I/O bits of the selected port are affected, regardless of their I/O mode and the state of IO/$\overline{M}$. The actual output level does not change until $\overline{IOW}$ returns high. (Glitch free output.)

A port can be read out when the latched Chip Enables are active and either $\overline{RD}$ goes low with IO/$\overline{M}$ high, or $\overline{IOR}$ goes low. Both input and output mode bits of a selected port will appear on lines AD_{0-7}.

To clarify the function of the I/O Ports and Data Direction Registers, the following diagram shows the configuration of one bit of PORT A and DDR A. The same logic applies to PORT B and DDR B.

8755A ONE BIT OF PORT A AND DDR A



231735-3

WRITE PA = ($\overline{IOW}$ = 0) • (CHIP ENABLES ACTIVE) • (PORT A ADDRESS SELECTED)
WRITE DDR A = ($\overline{IOW}$ = 0) • (CHIP ENABLES ACTIVE) • (DDR A ADDRESS SELECTED)
READ PA = {(IO/$\overline{M}$ = 1) • ($\overline{RD}$ = 0) + ($\overline{IOR}$ = 0)} • (CHIP ENABLES ACTIVE) • (PORT A ADDRESS SELECTED)

NOTE:
Write PA is not qualified by IO/$\overline{M}$.

Note that hardware RESET or writing a zero to the DDR latch will cause the output latch's output buffer to be disabled, preventing the data in the Output Latch from being passed through to the pin. This is equivalent to putting the port in the input mode. Note also that the data can be written to the Output Latch even though the Output Buffer has been disabled. This enables a port to be initialized with a value prior to enabling the output.

The diagram also shows that the contents of PORT A and PORT B can be read even when the ports are configured as outputs.

The erasure characteristics of the 8755A are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8755A in approximately 3 years while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 8755A is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8755A window to prevent unintentional erasure.

The recommended erasure procedure for the 8755A is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity x exposure time) for erasure should be a minimum of 15W-sec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000 μW/cm^2 power rating. The 8755A should be placed within one inch from the lamp tubes during erasure. Some lamps have a filter on their tubes and this filter should be removed before erasure.

PROGRAMMING

Initially, and after each erasure, all bits of the EPROM portions of the 8755A are in the "1" state. Information is introduced by selectively programming "0" into the desired bit locations. A programmed "0" can only be changed to a "1" by UV erasure.

The 8755A can be programmed on the Intel Universal Programmer (iUP), and iUPF8744A programming module.

The program mode itself consists of programming a single address at a time, giving a single 50 msec pulse for every address. Generally, it is desirable to have a verify cycle after a program cycle for the same address as shown in the attached timing diagram. **In the verify cycle (i.e., normal memory read cycle) 'V_{DD}' should be at +5V.**

System Interface with 8085AH

A system using the 8755A can use either one of the two I/O Interface techniques:

- Standard I/O
- Memory Mapped I/O

If a standard I/O technique is used, the system can use the feature of both CE_2 and $\overline{CE_1}$. By using a combination of unused address lines A_{11-15} and the Chip Enable inputs, the 8085AH system can use up to 5 8755A's without requiring a CE decoder. See Figure 4.

If a memory mapped I/O approach is used the 8755A will be selected by the combination of both the Chip Enables and IO/$\overline{M}$ using AD_{8-15} address lines. See Figure 3.

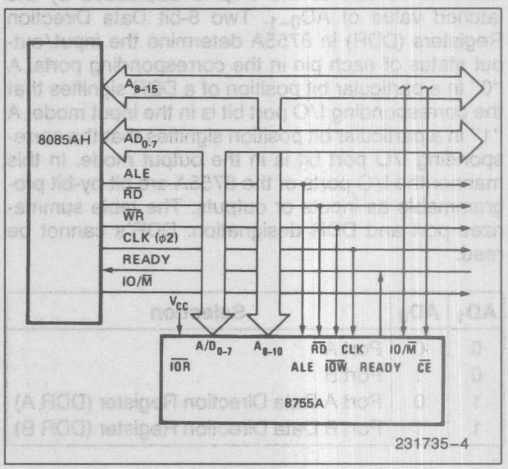


Figure 3. 8755A in 8085AH System (Memory-Mapped I/O)

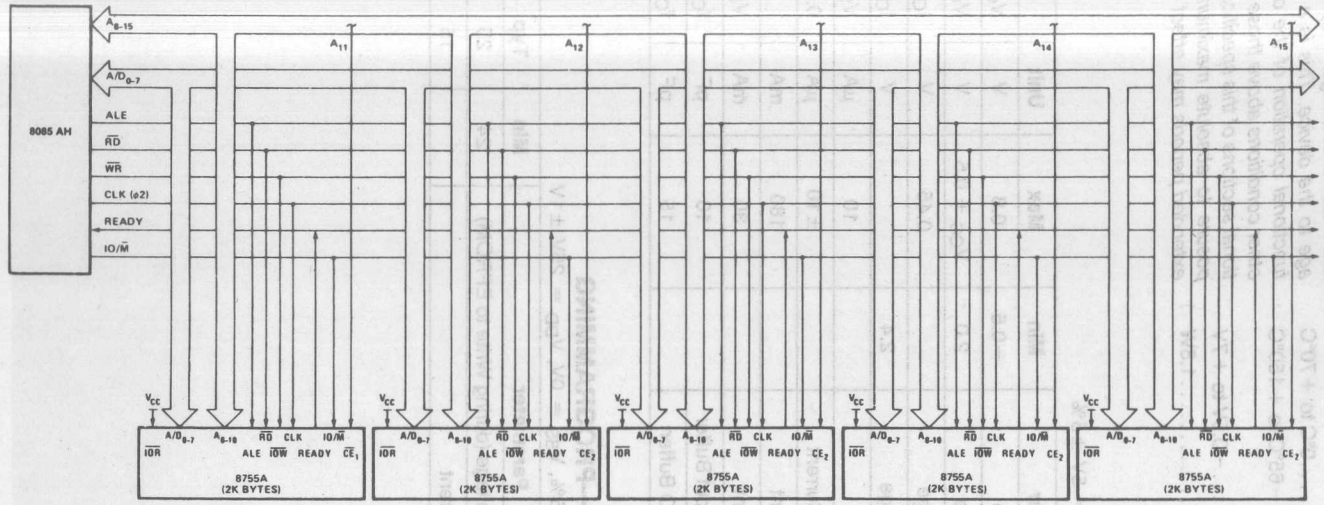


Figure 4. 8755A in 8085AH System (Standard I/O)

NOTE:
Use $\overline{CE_1}$ for the first 8755A in the system, and CE_2 for the other 8755A's. Permits up to 5-8755A's in a system without CE decoder.

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on any Pin with Respect to Ground . . . . . −0.5V to +7V
Power Dissipation 1.5W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = V_{DD} = 5V ±5%

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	V_{CC} = 5.0V
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.5	V	V_{CC} = 5.0V
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
I_{IL}	Input Leakage		10	μA	$V_{SS} \le V_{IN} \le V_{CC}$
I_{LO}	Output Leakage Current		±10	μA	$0.45V \le V_{OUT} \le V_{CC}$
I_{CC}	V_{CC} Supply Current		180	mA	
I_{DD}	V_{DD} Supply Current		30	mA	V_{DD} = V_{CC}
C_{IN}	Capacitance of Input Buffer		10	pF	f_C = 1 μHz
$C_{I/O}$	Capacitance of I/O Buffer		15	pF	f_C = 1 μHz

D.C. CHARACTERISTICS—PROGRAMMING

T_A = 0°C to 70°C, V_{CC} = 5V ±5%, V_{SS} = 0V, V_{DD} = 25V ±1V

Symbol	Parameter	Min	Typ	Max	Unit
V_{DD}	Programming Voltage (during Write to EPROM)	24	25	26	V
I_{DD}	Prog Supply Current		15	30	mA

A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ±5%

Symbol	Parameter	8755A		Unit
		Min	Max	
t_{CYC}	Clock Cycle Time	320		ns
T_1	CLK Pulse Width	80		ns
T_2	CLK Pulse Width	120		ns
t_f, t_r	CLK Rise and Fall Time		30	ns
t_{AL}	Address to Latch Set Up Time	50		ns
t_{LA}	Address Hold Time after Latch	80		ns
t_{LC}	Latch to READ/WRITE Control	100		ns
t_{RD}	Valid Data Out Delay from READ Control*		170	ns
t_{AD}	Address Stable to Data Out Valid**		450	ns
t_{LL}	Latch Enable Width	100		ns
t_{RDF}	Data Bus Float after READ	0	100	ns
t_{CL}	READ/WRITE Control to Latch Enable	20		ns
t_{CC}	READ/WRITE Control Width	250		ns
t_{DW}	Data in Write Set Up Time	150		ns
t_{WD}	Data in Hold Time after WRITE	30		ns
t_{WP}	WRITE to Port Output		400	ns
t_{PR}	Port Input Set Up Time	50		ns
t_{RP}	Port Input Hold Time to Control	50		ns
t_{RYH}	READY HOLD Time to Control	0	160	ns
t_{ARY}	ADDRESS (CE) to READY		160	ns
t_{RV}	Recovery Time between Controls	300		ns
t_{RDE}	READ Control to Data Bus Enable	10		ns

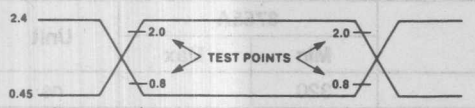
NOTES:
C_{LOAD} = 150 pF.
*Or $T_{AD} - (T_{AL} + T_{LC})$, whichever is greater.
**Defines ALE to Data Out Valid in conjunction with T_{AL}.

A.C. CHARACTERISTICS—PROGRAMMING

T_A = 0°C to 70°C, V_{CC} = 5V ±5%, V_{SS} = 0V, V_{DD} = 25V ±1V

Symbol	Parameter	Min	Typ	Max	Unit
t_{PS}	Data Setup Time	10			ns
t_{PD}	Data Hold Time	0			ns
t_S	Prog Pulse Setup Time	2			μs
t_H	Prog Pulse Hold Time	2			μs
t_{PR}	Prog Pulse Rise Time	0.01	2		μs
t_{PF}	Prog Pulse Fall Time	0.01	2		μs
t_{PRG}	Prog Pulse Width	45	50		ms

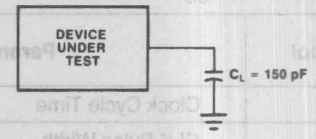
A.C. TESTING INPUT, OUTPUT WAVEFORM



231735-7

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing Measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT

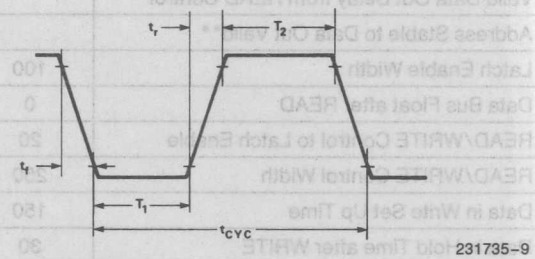


C_L = 150 pF
C_L Includes Jig Capacitance

231735-8

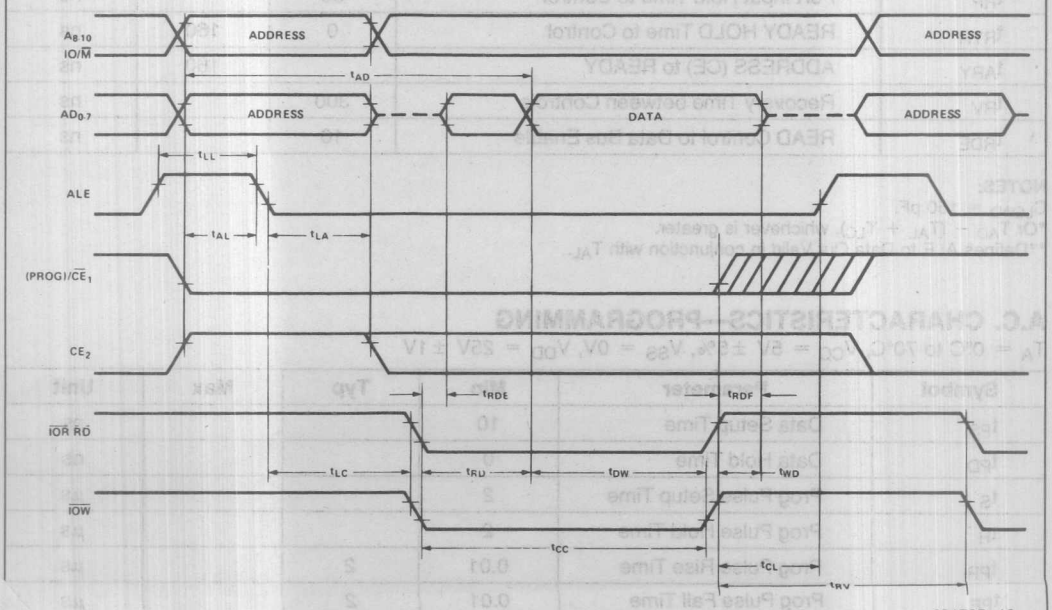
WAVEFORMS

CLOCK SPECIFICATION FOR 8755A



231735-9

PROM READ, I/O READ AND WRITE

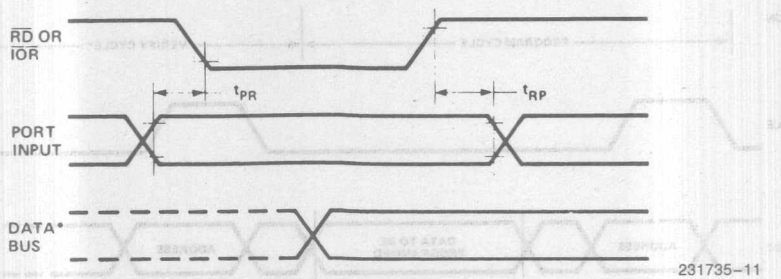


231735-10

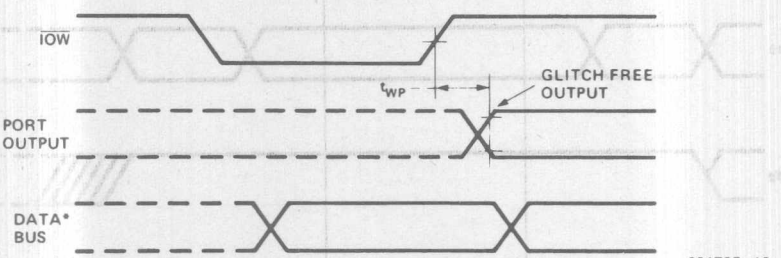
Please note that $\overline{CE_1}$ must remain low for the entire cycle.

WAVEFORMS (Continued)

I/O PORT

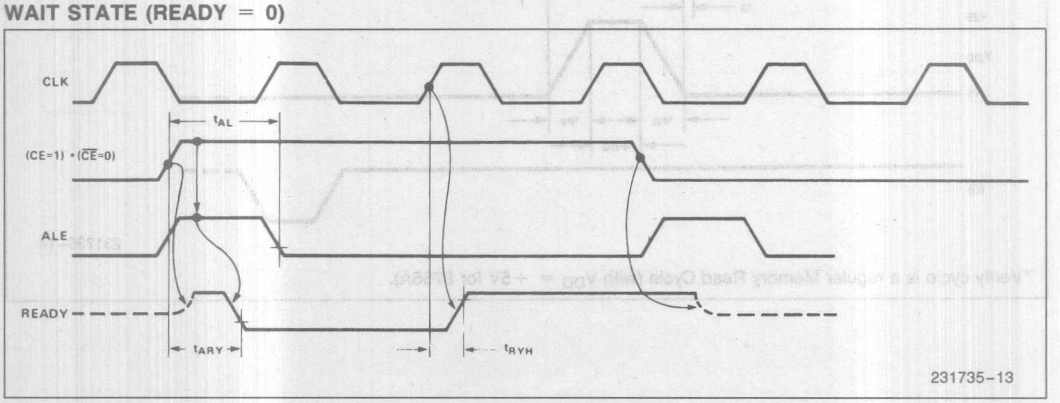


A. Input Mode



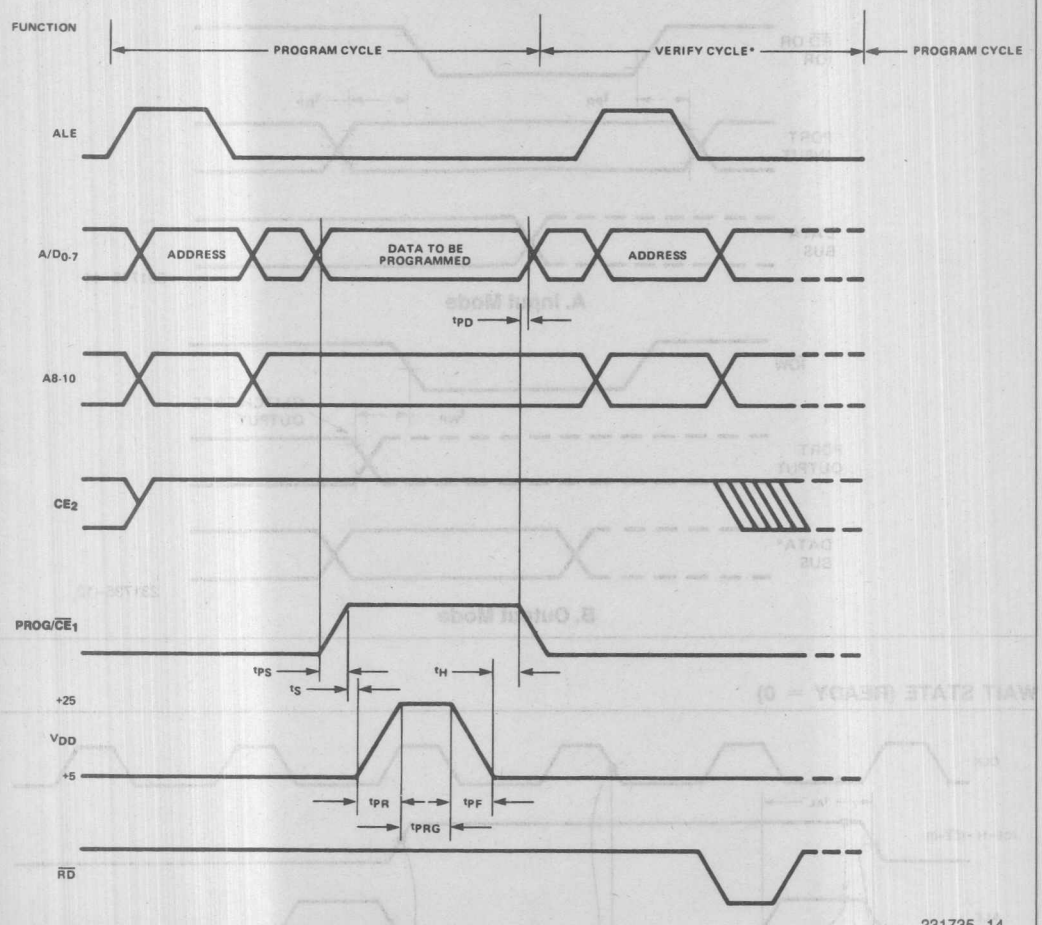
B. Output Mode

WAIT STATE (READY = 0)



WAVEFORMS (Continued)

8755A PROGRAM MODE



*Verify cycle is a regular Memory Read Cycle (with $V_{DD} = +5V$ for 8755A).